PENGEMBANGAN LANJUT WEB E-LEARNING CODEMANIAC MENGGUNAKAN WEB ADAPTIF BERBASIS LOG

SKRIPSI

Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Komputer

Disusun oleh:
Achmad Hanim Nur Wahid
NIM: 165150200111057

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

PENGESAHAN

PENGEMBANGAN LANJUT WEB E-LEARNING CODEMANIAC MENGGUNAKAN WEB ADAPTIF BERBASIS LOG

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Achmad Hanim Nur Wahid
NIM: 165150200111057

Skripsi ini telah diuji dan dinyatakan lulus pada
07 Juli 2021
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Fajar Pradana S.ST., M.Eng.	Fitra Abdurrachman Bachtiar, Dr. Eng., S.T., M. Eng.
NIP: 19871121 201504 1 004	NIP: 19840628 201903 1 006

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D
NIP: 19741118 200312 1 002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 11 Juni 2021

Achmad Hanim Nur Wahid

NIM: 165150200111057

PRAKATA

Puji Syukur kehadirat Allah SWT. yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga laporan skripsi yang berjudul "Pengembangan Lanjut Web E-Learning Codemaniac Menggunakan Web Adaptif Berbasis Log" ini dapat terselesaikan.

Penulis sepenuhnya sadar bawa laporan skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak yang telah memberikan dukungan berupa do'a, waktu, ilmu, bimbingan, arahan, serta motivasi. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Ibu Umi Hanik, Bapak M. Imron Rosyadi dan segenap keluarga besar atas seluruh dukungan, doa dan kasih sayang yang telah diberikan kepada penulis.
2. Bapak Fajar Pradana, S.ST., M.Eng., selaku dosen pembimbing I dan Bapak Dr. Eng. Fitra Abdurrachman Bachtiar, S.T., M. Eng. selaku Dosen pembimbing II yang telah memberikan bimbingan dan arahan yang sangat kami perlukan sehingga penulis dapat menyelesaikan skripsi ini.
3. Bapak Achmad Basuki, S.T., M.MG., Ph.D. selaku Ketua Jurusan Teknik Informatika dan Bapak Agus Wahyu Widodo, S.T., M.Cs. selaku Kepala Prodi Teknik Informatika.
4. Seluruh Ibu dan Bapak dosen Fakultas Ilmu Komputer Universitas Brawijaya atas segala ilmu, waktu, dan tenaga yang telah dibagikan kepada penulis.
5. Seluruh teman-teman penulis yang sudah bersedia meluangkan waktu dan pikiran untuk memberikan ilmu, saran, semangat serta dorongan yang sangat berpengaruh terhadap mental penulis dalam proses pengerjaan skripsi ini.

Penulis sepenuhnya sadar bahwa dalam penyusunan skripsi ini masih terdapat kekurangan, sehingga saran dan kritik yang membangun akan sangat bermanfaat bagi penulis. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 11 Juni 2021

Penulis

Hanim37@gmail.com

ABSTRAK

Achmad Hanim Nur Wahid, Pengembangan Lanjut *E-Learning* Codemaniac Menggunakan Web Adaptif Berbasis *Log*

Pembimbing: Fajar Pradana, S.ST., M.Eng. dan Dr. Eng. Fitra Abdurrachman Bachtiar, S.T., M. Eng.

Pendidikan merupakan salah satu kebutuhan yang sangat penting bagi peradaban manusia, terutama untuk usia anak hingga remaja. Kondisi pandemi yang saat ini sedang terjadi memaksakan Pendidikan menggunakan alternatif pembelajaran secara *online*. Codemaniac merupakan salah satu *e-learning* yang disusun menggunakan teknik *gamification* untuk meningkatkan motivasi pelajar. Namun, Codemaniac belum memiliki fitur adaptif yang dapat memaksimalkan perilaku setiap penggunannya. Untuk mengatasi masalah tersebut, pengembangan lanjut Codemaniac akan menambahkan fitur adaptif dengan memanfaatkan rekaman perilaku pengguna yang terdapat pada *log file*. Berdasarkan tingkah pengguna yang telah tercatat, akan dilakukan klusterisasi menggunakan algoritma *fuzzy c-means* sehingga akan terbentuk tiga macam *cluster*, setiap *cluster* akan menyajikan *user interface* yang berbeda. Sistem ini disusun dengan menggunakan SDLC *waterfall* dengan penambahan bahasa pemrograman *python*. Dalam pengembangannya menghasilkan tiga orang aktor, dengan tambahan lima kebutuhan fungsional dan satu kebutuhan non-fungsional. Untuk menguji sistem, digunakan metode *white box* untuk pengujian unit dan metode *black box* untuk pengujian valdiasi.

Kata kunci: *e-learning* adaptif, *log file*, *fuzzy cmeans*, *cluster*.

ABSTRACT

Achmad Hanim Nur Wahid, Pengembangan Lanjut *E-Learning* Codemaniac Menggunakan Web Adaptif Berbasis *Log*

***Supervisors*: Fajar Pradana, S.ST., M.Eng. dan Dr. Eng. Fitra Abdurrachman Bachtiar, S.T., M. Eng.**

Education is one of the most important needs for human civilization, especially for children and teenagers. The current pandemic condition is forcing education to use online learning alternatives. Codemaniac is one of the E-Learning which is builded using Gamification Techniques to increase student motivation. However, Codemaniac does not yet have adaptive features that can maximize the behavior of each user. To overcome this problem, further development of Codemaniac will add adaptive features by utilizing the recorded user behavior contained in the log file. Based on the recorded user behavior, clustering will be carried out using the fuzzy c-means algorithm so that three types of clusters will be formed, each cluster will present a different user interface. This system is compiled using SDLC waterfall with the addition of the Python programming language. In its development, it produces 3 actors, with an additional 5 functional requirements and 1 non-functional requirement. To test the system, the wthite box method is used for unit testing and the black box method is used for validation testing.

Keywords: adaptive e-learning, log file, fuzzy cmeans, cluster.

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
PRAKATA iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL x
DAFTAR GAMBAR xii
DAFTAR LAMPIRAN xiv
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 2
1.4 Manfaat 2
1.5 Batasan Masalah 3
1.6 Sistematika Penulisan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Kajian Pustaka 5
2.2 *E-Learning* 6
2.3 Web Adaptif 6
2.4 Perilaku Pengguna 7
2.5 Rekayasa Perangkat Lunak 8
2.5.1 *Software Development Life Cycle* (SDLC) 8
2.5.2 *Waterfall* 9
2.5.3 *Unified Modeling Language* (UML) 10
2.5.4 *Data Flow Diagram* 12
2.5.5 Pengujian Perangkat Lunak 14
2.6 Teknologi Pengembangan Sistem 17
2.6.1 *Log File* 17
2.6.2 Algoritma *Fuzzy C-Means* 17

BAB 3 METODOLOGI 20
3.1 Studi Literatur 20
3.2 Definisi Persyaratan 21
3.3 Perancangan Sistem 21
3.4 Implementasi 21
3.5 Integrasi dan Pengujian Sistem 22
3.6 Kesimpulan dan Saran 22
BAB 4 ANALISIS KEBUTUHAN 23
4.1 Gambaran Umum Sistem 23
4.1.1 Deskripsi Umum Sistem 23
4.1.2 Lingkungan Sistem 24
4.2 Identifikasi Aktor 24
4.3 Daftar kebutuhan fungsional 25
4.3.1 Aturan Penomoran 25
4.3.2 Kebutuhan Fungsional 25
4.3.3 Spesifikasi Kebutuhan Fungsional 26
4.4 Kebutuhan Non-Fungsional 27
4.5 Pemodelan Kebutuhan 27
4.5.1 *Use Case Diagram* 27
4.5.2 Use Case Scenario 27
BAB 5 PERANCANGAN DAN IMPLEMENTASI 30
5.1 Perancangan Sistem 30
5.1.1 Perancangan Arsitektur 30
5.1.2 Perancangan Komponen 35
5.1.3 Perancangan Basis Data 39
5.1.4 Perancangan Antarmuka 39
5.2 Implementasi Sistem 43
5.2.1 Spesifikasi Sistem 43
5.2.2 Implementasi Basis Data 43
5.2.3 Implementasi Kode Program 44
5.2.4 Implementasi Antarmuka 49
BAB 6 PENGUJIAN SISTEM 55

6.1 Pengujian Unit 55
6.1.1 Pengujian Unit *Method* Member 55
6.1.2 Pengujian Unit *Method* Cluster 58
6.1.3 Pengujian Unit *Script Clustering* 59
6.2 Pengujian Validasi 62
6.3 Pengujian Kompabilitas Aplikasi Web 66
BAB 7 PENUTUP 67
7.1 Kesimpulan 67
7.2 Saran 67
DAFTAR REFRENSI 68

DAFTAR TABEL

Tabel 2.1 Perilaku Pengguna 7
Tabel 4.1 Identikasi Aktor 24
Tabel 4.2 Penjelasan Aturan Penomoran Kebutuhan 25
Tabel 4.3 Kebutuhan Fungsional 25
Tabel 4.4 Kebutuhan Non - Fungsional 26
Tabel 4.5 Spesifikasi Kebutuhan Fungsional OOP 26
Tabel 4.6 Kebutuhan Non-Fungsional 27
Tabel 4.7 *Use case Scenario* Melihat daftar *cluster* 27
Tabel 4.8 *Use Case Scenario* Melakukan *Clustering* 28
Tabel 4.9 Use Case Scenario Menampilkan Dashboard 28
Tabel 4.10 *Use Case Scenario Login* 29
Tabel 4.11 *Use Case Scenario* Perubahan Antarmuka Pengguna 29
Tabel 5.1 Metode Perancangan Arsitektur Sistem 30
Tabel 5.2 *Pseudocode* Komponen *Method* Member 36
Tabel 5.3 *Pseudocode* Komponen *Method Clustering* 37
Tabel 5.4 *Pseudocode* Komponen *Script Clustering* 37
Tabel 5.5 Keterangan Gambar Perancangan Antarmuka *List Cluster* 40
Tabel 5.6 Keterangan Gambar Perancangan Antarmuka *Dashboard* 42
Tabel 5.7 Spesifikasi Perangkat Keras 43
Tabel 5.8 Spesifikasi Perangkat Lunak 43
Tabel 5.9 Implementasi Data Tabel *Cluster* 44
Tabel 5.10 Implementasi Kode Program *Method* Member 44
Tabel 5.11 Implementasi Kode Program *Method Clustering* 45
Tabel 5.12 Implementasi Kode Program *Proses Clustering* 46
Tabel 6.1 *Pseudocode Method* Member 55
Tabel 6.2 Hasil Pengujian Unit Method Member 57
Tabel 6.3 Pseudocode *Method Clustering* 58
Tabel 6.4 Hasil Pengujian Unit Method Cluster 59
Tabel 6.5 *Pseudocode Script Clustering* 59
Tabel 6.6 Hasil Pengujian Unit Script Clustering 62

Tabel 6.7 Pengujian Validasi *Use Case Scenario* Melihat Daftar *Cluster*..... 63
Tabel 6.8 Pengujian Validasi *Use Case Scenario* Melakukan *Clustering* 63
Tabel 6.9 Pengujian Validasi *Use Case Scenario* Menampilkan *Dashboard* 64
Tabel 6.10 Pengujian Validasi *Use Case Scenario Login*..... 64
Tabel 6.11 Pengujian Validasi Use Case Scenario Login Alternatif 1 65
Tabel 6.12 Pengujian Validasi Use Case Scenario Perubahan Antarmuka 65
Tabel 6.13 Pengujian Kompabilitas..... 66

DAFTAR GAMBAR

Gambar 2.1 Lapisan pada Rekayasa Perangkat Lunak 8
Gambar 2.2 Siklus Hidup Perangkat Lunak 9
Gambar 2.3 *Class Diagram* 11
Gambar 2.4 *Use Case Diagram* 11
Gambar 2.5 *Sequence Diagram* 12
Gambar 2.6 *Data Flow Diagram Level 0* 13
Gambar 2.7 *Data Flow Diagram Level* 1 13
Gambar 2.8 *Data Flow Diagram Level* 2 14
Gambar 2.9 Representasi Pengujian *Black Box* 15
Gambar 2.10 *Flowchart* 16
Gambar 2.11 *Flow Graph* 16
Gambar 3.1 Diagram Alur Metode Penelitian 20
Gambar 4.1 Arsitektur Sistem Codemaniac 23
Gambar 4.2 Kerangka Penelitian 24
Gambar 4.3 Aturan Penomoran Kebutuhan 25
Gambar 4.4 Use Case Diagram 27
Gambar 5.1 *Sequence Diagram* Melihat Daftar *Cluster* 30
Gambar 5.2 *Sequence Diagram* Melakukan *Clustering* 31
Gambar 5.3 Sequence Diagram Menampilkan *Dashboard* 31
Gambar 5.4 *Sequence Diagram Login* 32
Gambar 5.5 *Sequence Diagram* Perubahan Antarmuka Pengguna 33
Gambar 5.6 *Class Diagram* Pengembangan Codemaniac 34
Gambar 5.7 *Data Flow Diagram Clustering* Level 0 35
Gambar 5.8 *Data Flow Diagram Clustering* Level 1 35
Gambar 5.9 Perancangan *Physical Data Model* 39
Gambar 5.10 Perancangan Antarmuka *List Cluster* 40
Gambar 5.11 Perancangan Antarmuka Perubahan *User Interface* 41
Gambar 5.12 Perancangan Antarmuka *Dashboard* 42
Gambar 5.13 Implementasi Antarmuka Cluster 50
Gambar 5.14 Implementai Antarmuka *Dashboard* 51

Gambar 5.15 Implementasi Antarmuka Perubahan *Interface* 1 52
Gambar 5.16 Implementasi Antarmuka Perubahan *Interface* 2 52
Gambar 5.17 Implementasi Antarmuka Perubahan *Interface* 3 53
Gambar 5.18 Implementasi Antarmuka Perubahan *Interface* 4 53
Gambar 5.19 Implementasi Antarmuka Perubahan *Interface* 5 54
Gambar 5.20 Implementasi Antarmuka Perubahan *Interface* 6 54
Gambar 6.1 *Flowgraph Method* Member 56
Gambar 6.2 *Flowgraph Method Cluster* 58
Gambar 6.3 *Flowgraph Script Clustering* 61
Gambar 6.4 Hasil Pengujian Kompabilitas 66

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kondisi pandemi yang saat ini masih terjadi menjadi sebuah tantangan baru bagi sebuah negara untuk mengeluarkan kebijakan yang tepat guna menjaga kegiatan belajar mengajar tetap berjalan. Dalam sebuah berita yang diterbitkan pada *BBC News*, mereka melansir informasi dari UNESCO yang menampilkan kalau pada bulan April tahun 2020 terdapat 1,6 miliyar pelajar yang diliburkan dari sekolah serta universitas sebagai langkah-langkah buat menurunkan penyebaran Covid- 19. Angka tersebut merupakan 90% dari seluruh populasi siswa di Dunia (Hermansyah, 2020). Pemerintah kita telah mengambil langkah dengan menerapkan pembatasan sosial atau lebih dikenal dengan *social distancing*, imbasnya lembaga pendidikan harus melakukan proses belajar mengajar secara daring atau meliburkan sementara pelajar mereka.

Pendidikan *online* adalah proses belajar mengajar yang dilakukan dalam jaringan (daring) (Prof. Ir. Tian Belawati, 2020). Dari pengertian tersebut dapat dikatakan bahwa ada beberapa elemen yang perlu dipenuhi seperti perangkat keras yang memadai, jaringan internet yang stabil serta jumlah kuota internet yang cukup untuk dapat melaksanakan proses pembelajaran secara daring. Pembelajaran secara *online* dapat memiliki berbagai macam cara serta model yang dapat digunakan, asalkan tetap menerapkan *basic* yang dapat menunjang proses belajar mengajar yang berkualitas (Prof. Ir. Tian Belawati, 2020). Pembelajaran secara daring dengan menggunakan *e-learning* sangat bermanfaat bagi pelajar karena dapat diakses secara *online* kapan saja dan dari mana saja (Dhupia & Alameen, 2019).

Codemaniac merupakan sebuah sistem pembelajaran koding bahasa java berbasis web yang memerlukan koneksi internet untuk mengaksesnya. Codemaniac sendiri disusun oleh Dhanuari Bastari, mahasiswa Filkom Ub pada tahun 2017 dengan mengaplikasikan teknik *gamification* didalamnya. Teknik *gamification* sendiri tersusun dengan adanya fitur seperti menantang pengguna lain, fitur penghargaan untuk kondisi tertentu (badge), serta terdapat fitur *level* dan *experience* (Bastari, et al., 2017). Penggunaan teknik gamification ini diharapkan akan dapat meningkatkan nilai efisiensi, efektifitas, motivasi serta keterlibatan pelajar dalam *e-learning* (Urh, et al., 2015).

Penerapan *gamification* saja masih terbukti kurang dalam upaya meningkatkan motivasi belajar mahasiswa, untuk itu perlu adanya penerapan pendekatan secara *player-centric* yang memungkinkan sebuah sistem untuk menyesuaikan *gameplay* agar dapat sesuai dengan gaya belajar penggunanya (Priyambadha, et al., 2018). Agar supaya dapat memaksimalkan perilaku dari setiap pengguna, perlu adanya pembelajaran adaptif yang berdasarkan identifikasi gaya belajar individu merupakan fitur yang sangat penting untuk meningkatkan pembelajaran individu tersebut, terutama dalam pembelajaran *online* (Dhupia & Alameen, 2019).

Untuk mengembangkan *Codemaniac* menjadi diperlukan algoritma yang dapat menangkap perubahan minat dari pengguna yang dianalisis dalam rentan waktu tertentu sehingga dapat menciptakan personalisasi web dapat digunakan untuk menciptakan *user experience* yang sesuai untuk setiap pengguna (Das, et al., 2017). Algoritma fuzzy cmeans digunakan karena dapat digunakan untuk melakukan clustering dengan menggunakan data rekaman perilaku pengguna yang tercatat dalam *log file.* Data dalam log fie tersebut akan kemudian dibagi dalam beberapa *cluster* menggunakan algoritma *fuzzy cmeans* dengan memanfaatkan bahasa pemrograman *python*, nantinya setiap cluster akan memiliki tampilan *user interface* yang berbeda. Proses pengembangan ini akan menggunakan *Software Develepment Life Cycle Waterfall* serta menggunakan metode pengembangan berorientasi objek dan prosedural. Untuk mengembangkan web *e-learning Codemaniac* lebih lanjut lagi, maka disusunlah penelitian "Pengembangan Lanjut Web *E-Learning* Codemaniac Menggunakan Web Adaptif Berbasis *Log*" agar dapat mengembangkan web *e-learning* Codemaniac menjadi lebih baik.

1.2 Rumusan Masalah

Sesuai dengan penjabaran yang terdapat pada latar belakang, telah didapatkan rumusan masalah pada riset ini berupa:

1. Bagaimana hasil analisis kebutuhan Web *E-Learning* Codemaniac?
2. Bagaimana hasil rancangan Web *E-Learning* Codemaniac yang sesuai dengan hasil analisis kebutuhan perangkat lunak tersebut?
3. Bagaimana hasil implementaasi Web *E-Learning* Codemaniac yang sesuai dengan hasil rancangan kebutuhan perangkat lunak tersebut?
4. Bagaimana hasil pengujian Web *E-Learning* Codemaniac yang telah dilakukan?

1.3 Tujuan

Dengan disusunnya rumusan masalah, maka dapat ditentukan tujuan yang diharapkan dari riset ini sebagai berikut:

1. Menganalisis kebutuhan Web *E-Learning* Codemaniac.
2. Merancang Web *E-learning* Codemaniac yang sesuai dengan hasil analisis kebutuhan perangkat lunak tersebut.
3. Mengimplementasi Web *E-learning* Codemaniac yang sesuai dengan hasil rancangan kebutuhan perangkat lunak tersebut.
4. Menguji Web *E-learning* Codemaniac yang telah dihasilkan agar dapat dimanfaatkan sesuai dengan kebutuhan yang telah didefinisikan diawal.

1.4 Manfaat

Harapan yang tertuang dari riset ini yaitu menghasilkan tambahan fitur pada Web *E-Learning* Codemaniac yang dapat menyesuaikan kondisi pengguna sehingga dapat membantu para pengguna supaya lebih nyaman dan efektif saat

proses penggunaan. Selain itu diharapkan riset ini berguna sebagai acuan bagi riset kedepanya sehingga dapat diterapkan pada halaman web dibidang yang lainnya.

1.5 Batasan Masalah

Riset ini memiliki batasan seputar fitur perangkat lunak yang dihasilkan yaitu, codemaniac tetap berbasis website serta hanya bisa diakses oleh *device* dengan browser yang menunjang. Fokus riset ada pada pengembangan fitur web adaptif yang berbasis log, perubahan-perubahan yang diharapkan berfokus pada *user interface* dari *Web E-Learning* Codemaniac. Parameter yang digunakan unutk melakukan clustering adalah semua tingkah dalam log file, akan tetapi belum dapat ditentukan karakteristik untuk setiap cluster.

1.6 Sistematika Penulisan

Tata penulisan riset ini disusun supaya terdapat gambaran serta deskripsi dari laporan riset. Secara standar penulisan riset berisi beberapa bab sebagai berikut:

Bab I Pendahuluan

Bagian awal riset diisi dengan latar belakang pelaksaan riset, diikuti dengan rumusan masalah yang akan diteliti, lalu tujuan dan manfaat dari riset yang hendak dikantongi, setelahnya terdapat batasan masalah yang telah diatur dan ditutup dengan tata cara penulisan untuk Sistem *E-Learning* Codemaniac.

Bab II Landasan Kepustakaan

Bagian kedua diisi dengan acuan dan ilmu pengetahuan dasar mengenai pelaksaan penelitian yang mempunya relasi dengan perangkat lunak. Diikuti dengan prosedur yang digunakan untuk Sistem *E-Learning* Codemaniac.

Bab III Metodologi

Bagian ketiga diisi dengan literatur serta prosedur yang digunakan untuk melakukan riset, seperti analisis kebutuhan, teknik penggalian data, perancangan perangkat lunak, implementasi dan pengujian Sistem *E-Learning* Codemaniac.

Bab IV Analisa Kebutuhan

Bagian keempat diisi dengan analisa kebutuhan yang telah dilakukan untuk membangun rancangan sistem pada bab selanjutnya.

Bab V Perancangan dan Implementasi

Bagian kelima diisi dengan hasil perancangan yang telah disesuikan dengan hasil analisis kebutuhan yang telah didapatkan pada bab sebelumnya, dan berisi hasil implementasi perangkat lunak.

Bab VI Pengujian

Bagian keenam ini diisi dengan hasil dari proses pengujian dari perangkat lunak yang telah disusun.

Bab VII Penutup

Bagian ketujuh ini memuat kesimpulan dan saran terhadap pengembangan lanjut CodeManiac.

BAB 2 LANDASAN KEPUSTAKAAN

Bagian kedua berisi tentang ulasan teori yang hendak jadi dasar buat riset ini serta ulasan tentang teori-teori lain yang menunjang pengembangan fitur lunak. Sebagian teori dasar yang hendak dibahas pada bab kali ini seputar Rekayasa Fitur Lunak (RPL), *website* adaptif, *log file*, algoritma *fuzzy cmeans*, dan teori tentang pengujian perangkat lunak.

2.1 Kajian Pustaka

Riset ini dilaksanakan dengan menjajaki dari penelitian-penelitian yang telah ada sebelumnya. Rujukan yang diarahkan berperan sebagai pembagi batasan-batasan sistem sehingga bisa dikembangkan lagi dikemudian hari. Dengan bercermin pada rujukan yang ada, diharapkan pengembang bisa memperoleh sesuatu sistem baru yang belum terdapat pada riset sebelumnya.

Dhupian & Alameen (2019) dengan penelitian mereka yang berjudul "*Adaptive eLearning System : Conceptual Framework for Personalized Study Environment*". Penelitian tersebut mencatatkan bahwa penggunaan *e-learning* sangat bermanfaat bagi pelajar karena dapat diakses secara *online* tanpa batasan waktu dan tempat, namun sayangnya masih mudah dijumpai *e-learning* yang masih bersifat tradisional. Dalam hal ini artinya, *e-learning* belum menyediakan model pembelajaran yang lebih personal untuk setiap pelajar sehingga terdapat beberapa pelajar yang tidak cocok dengan *e-learning* tersebut. Berdasarkan permasalahhan tersebut, penelitian ini memberikan pengantar secara detail tentang konsep lingkungan *e-learning* yang adaptif beserta komponennya sehingga setiap pelajar dapat memilih cara belajar yang paling sesuai dengan dirinya. Dalam penelitian ini juga dijelaskan bahwa sistem yang adaptif adalah sistem yang terbentuk dari kombinasi bimbingan cerdas dan pembelajaran mesin.

Bayu, dkk (2018) dalam penelitiannya yang berjudul "Penggalian Perilaku Pemain Dalam Penentuan Tipe Permainan Pada *E-Learning* Pemrograman Berbasis *Gamification*" menyebutkan bahwa penerapan *gamification* saja masih terbukti kurang dalam upaya meningkatkan motivasi belajar mahasiswa, untuk itu perlu adanya penerapan pendekatan secara *player-centric* yang memungkinkan sebuah sistem untuk menyesuaikan *gameplay* agar dapat sesuai dengan gaya belajar penggunanya.

Das, dkk (2017) dalam penelitiannya yang berjudul "*Adaptive Web Personalization System Using Splay Tree*". Dalam penlitian ini telah dikembangkan sistem personalisasi web mutakhir yang dapat menangkap perubahan minat dari pengguna yang dianalisis dalam rentan waktu tertentu. Personalisasi web dapat digunakan untuk menciptakan *user experience* yang sesuai untuk setiap pengguna. Untuk menciptakan sistem personalisasi web tersebut, penelitian ini menggunakan *splay tree* yang merupakan struktur data mandiri bersifat adaptif sehingga dapat melacak perubahan pengguna.

2.2 *E-Learning*

Pembelajaran *online* atau bisa disebut dengan *e-learning* pada dasarnya adalah kegiatan belajar mengajar jarak jauh yang dilakukan dalam dan dengan menggunakan bantuan jaringan internet (Prof. Ir. Tian Belawati, 2020). Dengan fleksibilitas tersebut penggunaan *e-learning* sangat bermanfaat bagi pelajar karena dapat diakses secara *online* tanpa batasan tempat dan waktu (Dhupia & Alameen, 2019). Akan tetapi, sistem *e-learning* umumnya masih belum menganggap setiap pelajar bukan sebagai suatu individu tetapi hanya sebuah kelompok homogen (Tavangarian, et al., 2004). Seiring dengan perkembangan zaman, dalam dua dasawarsa terakhir kegiatan pendidikan atau pembelajaran melalui dan dengan bantuan teknologi internet atau pembelajaran *online* semakin berkembang (Prof. Ir. Tian Belawati, 2020). Perkembangan *e-learning* saat ini salah satunya mengarah pada pengembangan model pembelajaran yang lebih personal bahkan pelajar dapat memilih cara belajar yang paling sesuai untuk diri mereka sehingga pelajar tersebut merasa cocok dengan sistem pembelajaran yang diberikan (Dhupia & Alameen, 2019).

Saat ini telah banyak sekali *e-learning* yang dapat diakses, baik untuk golongan tertentu seperti untuk kampus ataupun *e-learning* yang dapat diakses oleh pengguna umum. Codemaniac adalah salah satu contoh *e-learning* milik FILKOM UB yang dapat diakses oleh banyak orang yang berisi pembelajaran pemrograman *java* dengan menerapkan metode *IOE-Behavior* dan *Gamification* (Bastari, et al., 2017). Selain itu juga terdapat *e-learning* seperti, *Udemy*, *Brain Academy*, *Educations*, dan Ruang Guru yang kesemuanya dapat diakses secara mudah melaui internet.

2.3 Web Adaptif

Seperti yang bisa kita pahami, *website* ialah sekumpulan halaman web yang silih berelasi antar halaman dan biasanya terletak pada *server* yang sama berisikan kumpulan data yang disediakan untuk keperluan tertentu. Kemajuan teknologi yang telah berkembang pesat telah berpengaruh pada banyak elemen, tidak terkecuali dalam hal Pendidikan, sistem Pendidikan yang dulunya bersifat *offline* mulai berkembang menjadi *online* berkat internet, salah satunya dengan menggunakan website atau biasa dikenal dengan sebutan *e-learning* (Dhupia & Alameen, 2019). Kemampuan sebuah web untuk dapat adaptif sangat penting dan meningkat dengan signifikan dalam konteks *Future Internet (FI)* yang mengharuskan web dapat beradaptasi secara mandiri dengan perubahan yang terjadi pada penyediaan layanan, ketersediaan barang dan konten, sumber daya komputasi dan konektivitas jaringan (Marquezan, et al., 2018). Seperti pada sistem penelurusan adaptif yang mempromosikan sebuah item dalam daftar hasil pencarian yang dianggap lebih relevan bagi pengguna sesuai dengan minat dan kebutuhan penggunan tersebut. (Brusilovsky, et al., 2007).

Sistem adaptif melakukan analisis informasi dan pola yang diperoleh dari setiap tingkah pengguna pada periode tertentu, akan tetapi informasi dan pola yang diperoleh mungkin akan berubah lagi dalam periode-periode setelahnya,

oleh karena itu perlu adanya pembaruan berkala dari informasi yang telah diekstraksi untuk menciptakan perubahan pada web (Das, et al., 2017) . Sistem adaptif dirancang untuk konteks penggunaan yang berbeda dan menjelajahi berbagai macam jenis personalisasi. Terdapat beberapa poin penting dalam penerapan sistem *e-learning* adaptif, poin pertama sistem dapat memahami gaya belajar masing-masing individu dan kemudian merumuskan rancangan pengajaran untuk setiap individu tersebut secara terpisah (Dhupia & Alameen, 2019).

2.4 Perilaku Pengguna

Pada penelitian ini perilaku pengguna adalah kegiatan-kegiatan yang telah pengguna kerjakan dalam sistem pembelajaran Codemaniac. Untuk mengembangkan sistem *e-learning* yang adaptif, maka sistem perlu menggali informasi yang dapat menampilkan tipe belajar pengguna, dengan cara melihat perilaku pengguna (Priyambadha, et al., 2018), tujuan pembelajaran pengguna, pola pembelajaran, tingkat pengetahuan dan bakat pengguna (Dhupia & Alameen, 2019). Penelitian ini menggunakan perilaku pengguna yang telah tersimpan dalam *log file* yang di dalamnya telah tercatat berbagai macam informasi perilaku pengguna, dapat dilihat pada Tabel 2.1. Sistem akan mencatat setiap kali pengguna meng-klik salah satu perilaku pada Tabel 2.1, jumlah klik pada setiap perilaku dan setiap pengguna akan dijumlah untuk menjadi nilai yang dapat digunakan untuk melakukan *clustering* dengan menggunakan algorima *fuzzy cmeans*. Jumlah klik memang hanya terdiri dari *request* HTTP(S) yang sesuai dengan fitur yang diklik oleh pengguna tetapi dapat secara langsung menunjukkan kekhawatiran, niat, dan tindakan pengguna (Fei, et al., 2021).

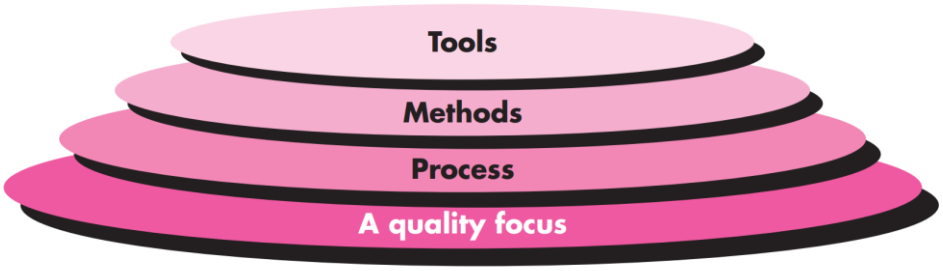
Tabel 2.1 Perilaku Pengguna

No.	Perilaku Pengguna
1.	*Accept request friend*
2.	*Check exercise*
3.	*Compile exercise*
4.	*Confirm challenge*
5.	*Detail exercise*
6.	*Login*
7.	*Logout*
8.	*Search friends*
9.	*Send request friend*
10.	*Show profile*
11.	*Show challenge*

12.	*Show exercise*
13.	*Show friends*
14.	*Submit exercise*
15.	*Take challenge*
16.	*Take course*
17.	*Take exercise*
18.	*Test exercise*

2.5 Rekayasa Perangkat Lunak

Menilik dari penjelasan dari Roger S. Pressman, rekayasa perangkat lunak adalah teknologi yang bersusun, seperti pada Gambar 2.1, pendekatan-pendekatan rekayasa (tidak terkecuali rekayasa perangkat lunak) diawali dengan membangun pondasi pada fokus kualitas, setelahnya berfokus pada lapisan proses yang memiliki fungsi untuk mendefinisikan kerangka kerja yang harus ditetapkan yang didalamnya berisi dasar untuk pengendalian manajemen proyek perangkat lunak, lapisan berikutnya adalah metode yang berisikan petunjuk teknis tentang pembuatan perangkat lunak, dan lapisan yang terakhir adalah alat yang didalamnya menyediakan dukungan *semi-automatic* atau *full-automatic* untuk lapisan proses dan metode (Pressman, 2010).



Gambar 2.1 Lapisan pada Rekayasa Perangkat Lunak

(Sumber : Roger S. Pressman, 2010)

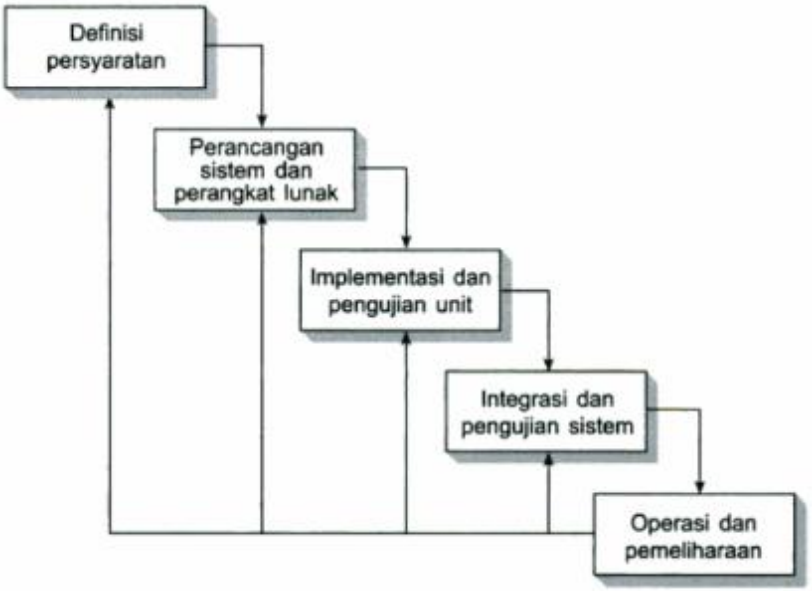
2.5.1 *Software Development Life Cycle* (SDLC)

Perkembangan awal pengembangan perangkat lunak cuma suatu pekerjaan untuk menuliskan kode-kode program oleh seseorang *programmer* guna membongkar permasalahan. Seakan mengikuti perjalanan waktu, kebutuhan sistem bertambah besar serta lingkungan memerlukan kedudukan ekstra seperti perancang, analis, *programmer* dan *user* guna menyusun sebuah sistem. Supaya bisa berjalan dengan baik, lahirlah pelbagai pendekatan sistem guna meningkatkan sistem yang bisa diketahui sebagai *System Development Life Cycle* (SDLC).

Supaya dapat berhasil menyusun sebuah sistem perangkat lunak peran *Software Development Life Cycle* (SDLC) sangatlah kritikal, harus memilih SDLC yang tepat dengan kebutuhan yang ditetapkan. Dalam mengembangkan perangkat lunak ada banyak jenis model yang dapat digunakan. Jenis model yang akan disebutkan memiliki metodologi populer dan baru, contohnya *prototyping*, *waterfall*, *incremental*, *spiral development*, *extreme programming* dan *rapid application development*. Sebuah model proses untuk suatu perangkat lunak dipilih berdasarkan karakteristik projek atau aplikasi tersebut (Pressman, 2010).

2.5.2 *Waterfall*

Software Development Life Cycle (SDLC) tipe *Waterfall* merupakan model yang mengambil aktivitas proses awal semacam spesifikasi, pengembangan, validasi, serta evolusi setelah itu mempresentasikannya selaku tingkatan-tingkatan proses yang berbeda semacam spesifikasi persyaratan, perancangan perangkat lunak, implementasi, pengujian serta seterusnya (Sommerville, 2011). Langkah-langkah dari teknik *waterfall* ini dapat dilihat pada Gambar 2.2 yang memetakan kegiatan-kegiatan pengembangan dasar, yaitu :



Gambar 2.2 Siklus Hidup Perangkat Lunak

(Sumber : Sommerville, 2011)

Dari gambar 2.2 dapat kita lihat terdapat beberapa tahapan yang ada pada model *waterfall* yang harus dilakukan secara berurutan. Untuk penelitian yang dilakukan saat ini, tahapan-tahapan yang dilalui akan berakhir pada langkah integrasi dan pengujian sistem dan tidak meyertakan tahapan terakhir yaitu, operasi dan pemeliharaan. Penjelasan dari setiap langkah akan dijelaskan dibawah ini:

1. Analisis dan definisi persyaratan menjadi tahapan awal ini akan menjadi dasar acuan untuk langkah-langkah pengerjaan sistem selanjutnya. Dalam fase ini terdapat seluruh kebutuhan, batasan dan tujuan sistem yang

ditentukan melalui komunikasi yang melibatkan dari kedua belah pihak. Persyaratan ini selanjutnya dicatat serta dijabarkan dengan mendetail untuk mudah dipahami dan digunakan untuk spesifikasi sistem yang akan disusun.

2. Perancangan sistem dan perangkat lunak merupakan langkah kedua untuk ditentukannya desain sistem secara utuh baik dari persyaratan untuk perangkat lunak maupun perangkat keras. Perancangan ini membutuhkan pengenalan dan penjelasan tentang sistem yang jelas beserta relasinya. *Unified Modelling Language* (UML) digunakan pada langkah selanjutnya sebagai bahasa yang digunakan untuk melakukan pemodelan secara grafis sesuai dengan hasil dari analisis dari langkah sebelumnya. Secara umum, pemodelan ini bertujuan sebagai definisi dari ruang lingkup sistem yang akan disusun serta berisi hubungan yang ada dalam sistem secara utuh.
3. Implementasi dan pengujian unit pada langkah ketiga. Bertujuan untuk membentuk hasil perancangan perangkat lunak yang telah disetujui pada langkah sebelumya sehingga dapat terealisasikan menjadi sebuah program. Dalam langkah ini juga dilakukan verifikasi bahwa program sudah memenuhi kebutuhan sistem yang telah ditentukan atau masih belum terpenuhi secara keseluruhan.
4. Integrasi dan pengujian sistem pada langkah keempat menjadi wadah untuk melakukan integrasi serta langkah untuk melakukan pengujian terhadap program. Program perlu diverifikasi pada fase sebelumnya untuk selanjutnya mengintegrasi dan menguji program sebagai sebuah sistem yang utuh.
5. Operasi dan pemeliharaan pada langkah kelima mencakup perbaikan dari pelbagai kekurangan yang terlewati dari fase-fase sebelum ini, perbaikan atas fase implementasi sistem dan pengembangan fitur-fitur lainnya pada sistem.

2.5.3 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah proses mendefinisikan komponen yang akan digunakan untuk membangun sebuah sistem dan untuk menggabungkan semua komponen-komponen *interface* yang ada (Pressman, 2001). UML sendiri disusun sebagai gambaran dari sebuah mekanisme sistem yang berisi beberapa model diagram untuk membantu mempermudah pemahaman terhadap sistem. UML tersusun dari sejumlah model diagram, seperti:

2.5.3.1 Class Diagram

Sistem yang menggunakan pendekatan *object oriented* akan menggunakan *class diagram* guna menyatakan kelas-kelas yang ada serta relasi yang dimiliki antar kelas tersebut. Pembuatan *class diagram* dapat dilakukan dengan memberi nama kelas ke dalam sebuah kotak. Selanjutnya beri relasi antar kelas dengan menambahkan garis di antara kelas-kelas tersebut. Seperti pada Gambar 2.3 sebuah *class diagram* yang sederhana menampilkan dua kelas

yang diberi nama *Patient* dan *Patient Record* serta saling terhubung (Sommerville, 2011).

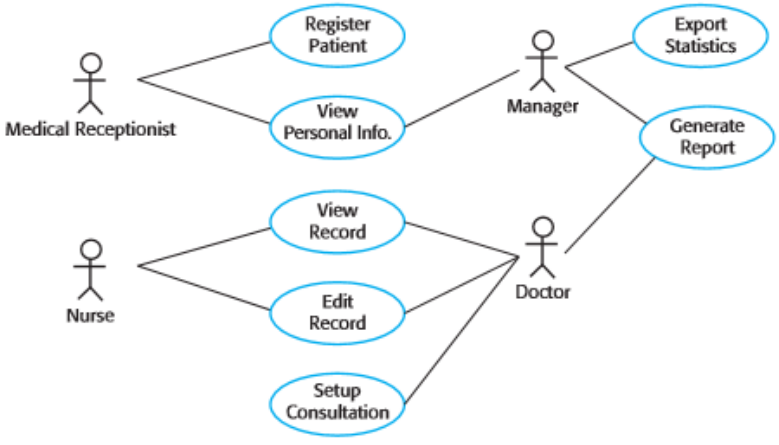
Gambar 2.3 *Class Diagram*
(Sumber : Sommerville, 2011)

Pada tingkatan *class diagram* yang lebih mendalam, informasi yang terdapat pada kelas tersebut harus dimasukkan pada dalam kotak *class diagram*. Misalkan, *object Patient* mempunyai atribut *Address* kemudian pada *class diagram* yang sama harus mempunyai operasi yang bernama *ChangeAddress*. Seperti pada contoh yang ada pada Gambar 2.3, yaitu :

a. Nama dari object selalu ada pada bagian teratas.
b. Atribut dari kelas terletak pada bagian tengah. Bagian ini juga harus mencantumkan nama atribut serta tipenya.
c. Operasi yang terletak pada bagian paling bawah.

2.5.3.2 Use Case Diagram

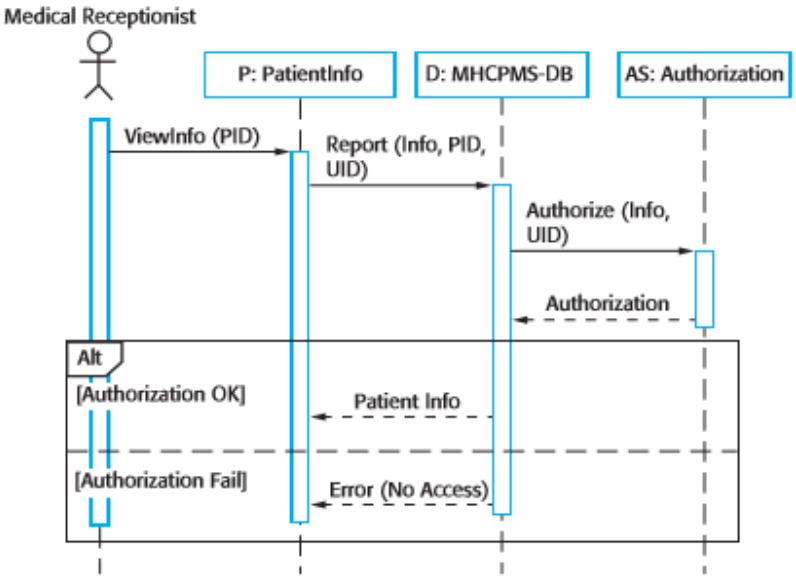
Teknik untuk mengumpulkan kebutuhan dalam *objectory method* adalah *Use Case Diagram* (Jacobson et al., 1993). Teknik ini sudah menjadi fitur yang sangat mendasar dalam UML. *Use Case* menjadi penunjuk aktor yang termasuk dalam interaksi dan jenis tipe dari interaksi tersebut. Kemudian interaksi-interaksi tersebut dijelaskan pada sebuah informasi yang lebih mendetail tentang relasi antara aktor dengan sistem tersebut. Dapat dilihat pada Gambar 2.4 yang menampilkan sebuah *Use Case* untuk *patient information system* (Sommerville, 2011).



Gambar 2.4 *Use Case Diagram*
(Sumber : Sommerville, 2011)

2.5.3.3 Sequence Diagram

Sequence diagram pada UML berguna untuk memodelkan interaksi antara aktor dengan objek pada sistem serta interaksi antara objek dan objek yang ada dalam sistem tersebut. Gambar 2.5 menunjukkan model diagram interaksi yang ada antara *View Patient Information use case.* Pada diagram tersebut informasi yang dimiliki pasien dapat dilihat oleh *medical receptionist* (Sommerville, 2011).

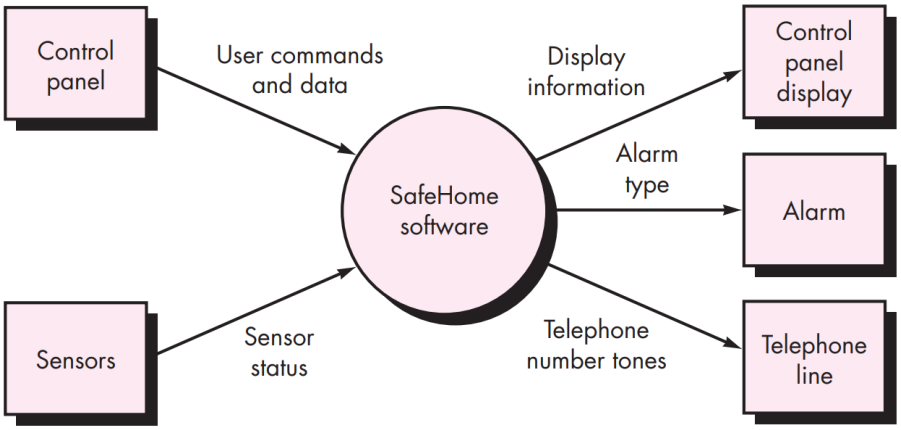


Gambar 2.5 *Sequence Diagram*

(Sumber : Sommerville, 2011)

2.5.4 *Data Flow Diagram*

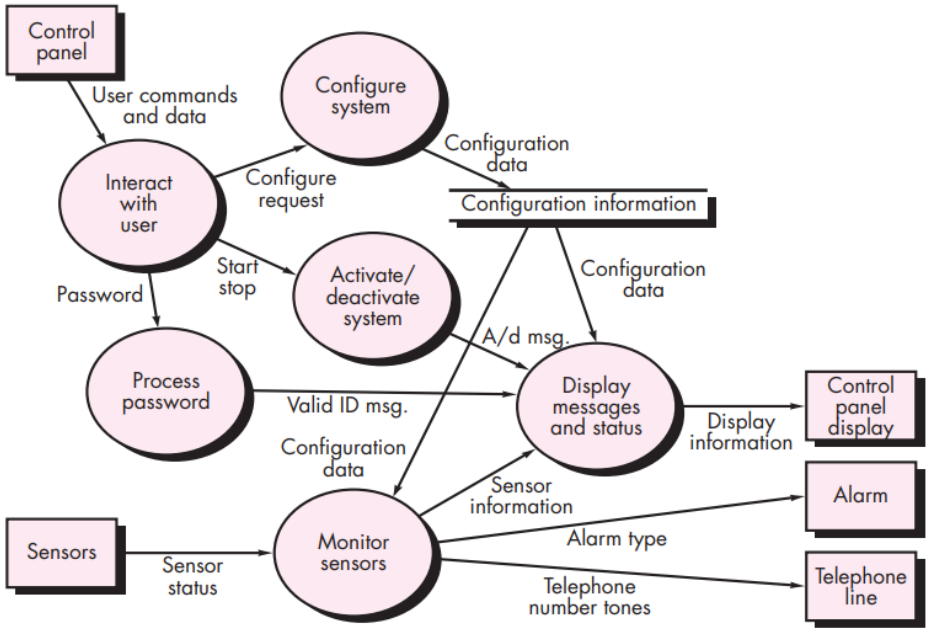
Data flow diagram (DFD) menggambarkan tampilan *input-proccess-output* dari suatu sistem, objek data mengalir kedalam sistem kemudian diolah sedemikian rupa sehingga menghasilkan keluaran tertentu (Pressman, 2010). *Data flow diagram* disusun untuk menunjukkan alur dan transformasi data melalui sistem (Singh, n.d.). *Data flow diagram* tidak termasuk dalam UML tetapi dapat dipakai untuk menambah wawasan tentang alur sistem (Pressman, 2010). Dalam penerapannya, *data flow diagram* dapat menghasilkan beberapa tingkat level, mulai dari 0-4. Gambar 2.6 merupakan contoh data flow diagram level 0.



Gambar 2.6 *Data Flow Diagram Level 0*

(Sumber : Roger S. Pressman, 2010)

Pada level 0 digambarkan tentang sistem sebagai satu gelembung serta diiringi dengan masukan dan keluaran primer. Entitas eksternal primer digambarkan dalam simbol persegi sebagai tempat mengeluarkan dan menyimpan data. Panah berlabel berfungsi untuk mewakili objek atau data atau hierarki objek data. Meningkat menuju DFD level 1, gelembung *SafeHome Software* pada Gambar 2.6 dipecah menjadi beberapa gelembung yang berisi proses-proses yang lebih sederhana seperti pada Gambar 2.7.

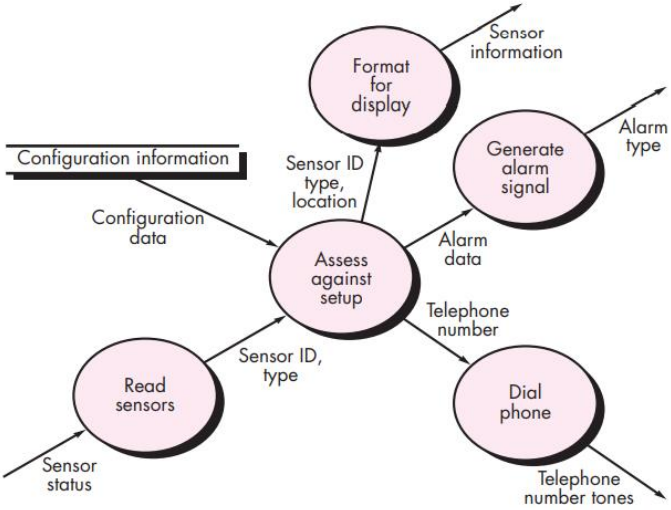


Gambar 2.7 *Data Flow Diagram Level* 1

(Sumber : Roger S. Pressman, 2010)

Perlu diperhatikan jumlah elemen masukan dan keluaran primer serta arah *flow* yang digambarkan dengan tanda panah pada level 0. Proses yang direpresentasikan pada DFD level 1 dapat ditingkatkan lagi ke level yang lebih

rendah, seperti pada Gambar 2.8. Proses sensor monitor pada level 1 dipecah lagi menjadi level 2 pada Gambar 2.8. pengembangan DFD ini dapat berlanjut hingga setiap gelembung menjalankan fungsi yang sederhana (Pressman, 2010).



Gambar 2.8 *Data Flow Diagram Level* 2

(Sumber : Roger S. Pressman, 2010)

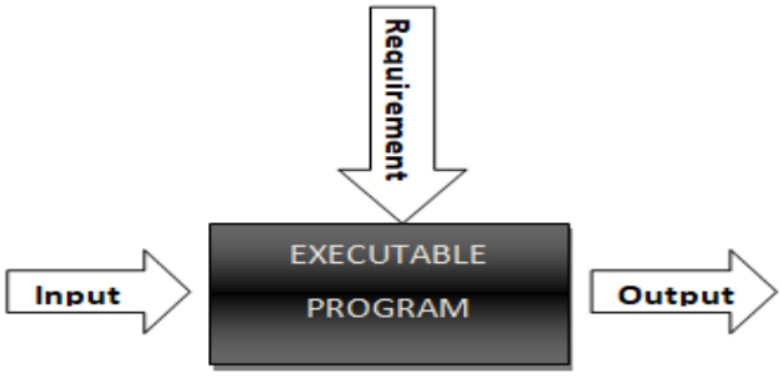
2.5.5 Pengujian Perangkat Lunak

Pengujian perangkat lunak memilik tujuan sebagai penjamin kualitas program yang diuji. Selain itu juga bermanfaat sebagai penyaji kajian pokok dari spesifikasi, desain dan pengodean. Untuk memenuhi kebutuhan pengguna dan *stakeholder* yang menjadi pengguna sistem maka pengujian perlu dilakukan (Sommerville, 2011).

2.5.5.1 Pengujian Black Box

Pengujian yang menitik beratkan pada persyaratan fungsional perangkat lunak biasa disebut dengan pengujian *black box* (Pressman, 2010). Disebut dengan pengujian *black box* karena penguji tidak perlu mengetahui tentang implementasi *source code* pada aplikasi tersebut (Verma, et al., 2017). Pengujian *black box* ini memungkinkan para penguji untuk menciptakan suatu skema yang dapat dikerjakan oleh semua fungsi dalam perangkat lunak tersebut (Bastari, et al., 2017).

Terdapat beberapa macam metode pengujian *black box*, salah satunya adalah *functional testing* yaitu, proses pengujian pada suatu fungsi atau fitur dari sebuah sistem dengan hanya melibatkan observasi nilai keluaran pada nilai masukan tertentu. Pada Gambar 2.6 dapat dilihat representasi dari pengujian *black box*. Dimisalkan *requirement* yang ditentukan adalah sistem dapat melakukan *login* dengan *username* dan *password* sebagai masukan kemudian halaman *dashboard* sebagai keluaran jika masukan benar.



Gambar 2.9 Representasi Pengujian *Black Box*

(Sumber : Verma, dkk, 2017)

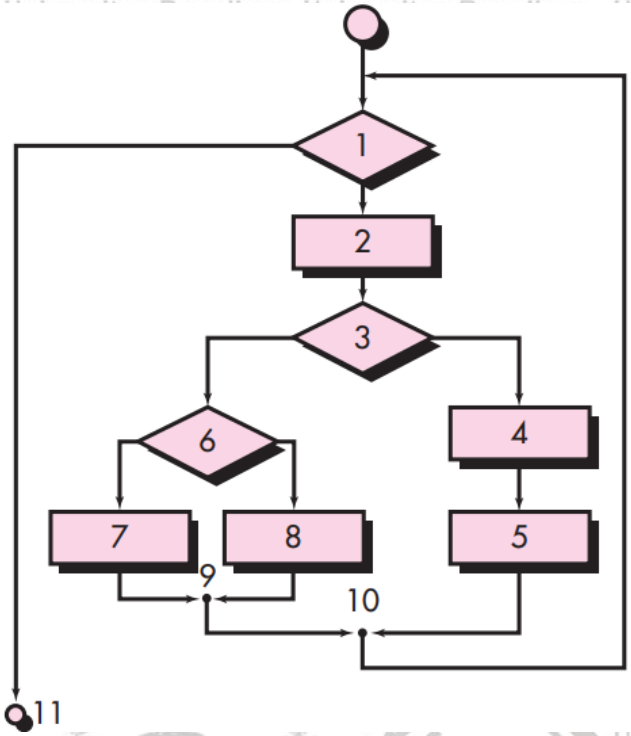
2.5.5.2 Pengujian White Box

Pengujian *white box* dapat diartikan sebagai pengujian yang menggunakan *source code* pada sistem untuk menguji *internal logic*, *code structure* dan *control flow* dari sebuah sistem (Verma, et al., 2017). Tujuan dari pengujian *white box* adalah untuk memastikan bahwa *source code* atau algoritma sistem sudah berjalan dengan benar (Bastari, et al., 2017). Untuk melakukan pengujian *white box*, penguji harus memiliki pengetahuan serta kemampuan tentang *source code* sistem tersebut (Verma, et al., 2017).

Basis path testing merupakan salah satu teknik yang termasuk dalam kelompok pengujian *white box*. Metode ini bertujuan unutk mengetahui nilai kompleksitas logika (*cyclomatic complexity*) dari sebuah program (Pressman, 2010). Nilai kompleksitas logika tersebut didapatkan dari *flow graph* suatu *source code* atau sebuah algoritma, dan digunakan untuk pondasi dasar dalam membuat jalur eksekusi program. Jalur eksekusi ini nantinya akan digunakan sebagai dasar untuk pembangunan kasus uji untuk menjalankan pernyataan-pernyataan pada program dalam satu kali fase pengujian (Bastari, et al., 2017). Menurut Roger S. Pressman (2010) terdapat empat elemen pengujian, yaitu:

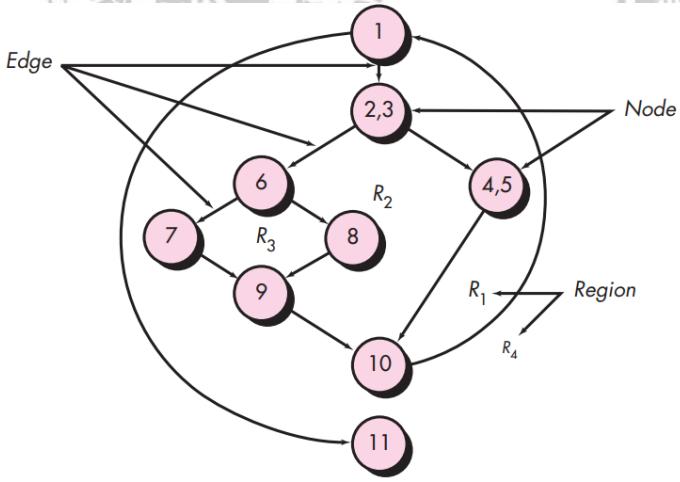
1. *Flow Graph*

   Untuk menggambarkan penggunaan *flow graph*, perhatikan Gambar 2.7 yang merupakan sebuah *flowchart* dan Gambar 2.8 yang merupakan sebuah *flow graph.* Mengacu pada Gambar 2.7, setiap lingkaran disebut *node* menggantikan satu atau mungkin lebih pernyataan prosedural, untuk *node* yang menggantikan suatu kondisi disebut dengan *predicate node*. Setiap *node* tersebut dihubungkan dengan tanda panah atau biasa disebut dengan *edge*. *Edge* sendiri adalah penggambaran arah jalan suatu algoritma program pada *flowchart*, daerah yang berada diantara *edge* dan *node* disebut *region*. Untuk menghitung jumlah *region*, daerah yang berada diluar dari *flow graph* akan dinilai sebagai satu *region* (Pressman, 2010).



Gambar 2.10 *Flowchart*

(Sumber : Roger S. Pressman, 2010)



Gambar 2.11 *Flow Graph*

(Sumber : Roger S. Pressman, 2010)

2. Cyclomatic Complexity

Cyclomatic complexity V(G) ialah metrik perangkat lunak yang dapat memberikan nilai kuantitatif dari kompleksitas logis suatu program, semakin tinggi nilai yang didapatkan maka semakin sulit program untuk dipahami dan diuji. Untuk mendapatkan tersebut dapat dihitung dengan tiga cara, yaitu:

a. V(G) = Jumlah *region* dalam *flow graph*
b. V(G) = E (Edge) – N (Node) + 2
c. V(G) = P (Predicate Node) + 1

3. Independent Path

Semua jalur yang dilewati program hingga menghasilkan keluaran tertentu disebut dengan *independet path*, setiap *independent path* seminimal mungkin telah melewati satu buah edge yang belum pernah dilewati oleh *independent path* sebelumnya.

4. Test Case

Dalam melakukan pengujian diperlukan skema data masukan guna mengecek alur logika atau kondisi pada setiap *independent path*.

2.6 Teknologi Pengembangan Sistem

2.6.1 *Log File*

Dalam komputasi, *file log* adalah *file* yang merekam peristiwa yang terjadi dalam sistem operasi atau perangkat lunak lainnya saat sedang berjalan atau pesan antara pengguna yang berbeda dari perangkat lunak komunikasi. Meskipun mereka dapat berisi berjumlah hal, pada umumnya *file log* digunakan untuk menampilkan semua peristiwa yang terkait dengan sistem atau aplikasi. Misalnya, pada *backup file* yang menyimpan *file-file log* yang menunjukkan dengan tepat apa yang telah terjadi (atau tidak terjadi) selama penggunaan aplikasi sebelum dicadangan. Meskipun sebagian besar *file log* berbentuk ekstensi *file.log*, beberapa aplikasi juga dapat menggunakan *file.txt*.

Dalam penelitian ini, telah terdapat data yang tersimpan dalam sebuah tabel dalam *database* yang berisi pelbagai tingkah yang telah *user* lakukan. Tingkah-tingkah tersebut terekam secara otomatis saat *user* menggunakan suatu fungsi yang ada dalam sistem. Nantinya kumpulan tingkah-tingkah *user* yang telah tersimpan tadi akan diolah sedemikian rupa sehingga akan menghasilkan pola yang berbeda dari tiap *user* yang kemudian dapat dikelompokkan dalam beberapa *cluster*.

2.6.2 Algoritma *Fuzzy C-Means*

Algoritma ini awalnya dikembangkan oleh Dunn pada tahun 1973 dan selanjutnya diperbaiki oleh Bezdek pada tahun 1981. Algoritma FCM merupakan teknik untuk melakukan *clustering* dengan pendekatan *fuzzy*, artinya setiap data yang akan dikelompokkan sangat mungkin menjadi anggota dari kelompok lainnya (James C. Bezdek, 1984). Alogitma FCM digunakan untuk mendapatkan titik tengah *cluster* namun pada awal pengondisian, titik tengah *cluster* pada algoritma masih jauh dari kata akurat, serta setiap data akan memiliki derajat keanggotaan untuk tiap-tiap *cluster*. Perlu adanya iterasi untuk mendapatkan nilai akurasi titik tengah *cluster* dan nilai keanggotaan tiap data yang sesuai

dengan fungsi objek sehingga akan dapat digambarkan jarak antara titik data dengan pusat *cluster*. Algoritma *fuzzy c-means* membagi suatu gugus data ke dalam *cluster* yang telah ditentukan sebelumnya sesuai dengan kebutuhan. Oleh sebab itu, perlu adanya parameter untuk memilih jumlah *cluster* optimal dalam data, umumnya disebut sebagai masalah *cluster validation* (Zakaria, 2008). Berikut adalah algoritma *clustering* FCM:

1. Masukan data yang hendak diproses pada *cluster* X, berupa matriks yang berukuran n x p (n = jumlah sampel data, p = atribut pada setiap data). X_{kj} = data sampel ke-k (k = 1, 2, …, n), atribut ke-j (j = 1, 2, …, m).
2. Tentukan:
   a. Iterasi awal : $t = 1$
   b. *Error* terkecil yang diharapkan : ξ
   c. Fungsi objektif awal : $P_0 = 0$
   d. Pangkat pembobot : m
   e. Jumlah *cluster* : c
   f. Iterasi maksimal : MaxIter
3. Bangkitkan bilangan random (μ_{ik}, i = 1,2, … , c; k = 1,2, … , n), sebagai elemen-elemen matriks partisi awal U

$$\boldsymbol{U_0} = \begin{bmatrix} \mu_{11}(X_1) & \mu_{12}(X_2) & \cdots & \mu_{1c}(X_c) \\ \mu_{21}(X_1) & \mu_{22}(X_2) & \cdots & \mu_{2c}(X_c) \\ \vdots & \vdots & & \vdots \\ \mu_{n1}(X_1) & \mu_{n2}(X_2) & \cdots & \mu_{nc}(X_c) \end{bmatrix}$$

Pada *fuzzy clustering*, matriks partisi harus dapat memenuhi kondisi sebagai berikut:

$$\mu_{ik} = [0,1]; (1 \le i \le c; 1 \le k \le n)$$

$$\sum_{i=1}^{n} \mu_{ik} = 1; 1 \le i \le c$$

$$0 < \sum_{i=1}^{c} \mu_{ik} < c;\ 1; 1 \le k \le n$$

Hitung jumlah setiap kolom (atribut):

$$Q_j = \sum_{i=1}^{c} (\mu_{ik})$$

Dengan j = 1, 2, …, m

Kemudian hitung:

$$\mu_{ik} = \frac{\mu_{ik}}{Q_j}$$

4. Hitung pusat *cluster* ke-K: V_{ij}, dimana i = 1, 2, …, c dan j = 1, 2, …, m:

$$V_{ij} = \frac{\sum_{k=1}^{n}\left((\mu_{ik})^{m} * X_{kj}\right)}{\sum_{k=1}^{n}(\mu_{ik})^{m}}$$

$$V = \begin{bmatrix} v_{11} & \cdots & v_{1m} \\ \vdots & \ddots & \vdots \\ v_{c1} & \cdots & v_{cm} \end{bmatrix}$$

5. Hitung fungsi objektif pada iterasi ke-t, P_t dengan menggunakan persamaan sebagai berikut:

$$P_t = \sum_{k=1}^{n}\sum_{i=1}^{c}\left(\left[\sum_{j=1}^{m}\left(x_{kj} - v_{ij}\right)^2\right](\mu_{ik})^{m}\right)$$

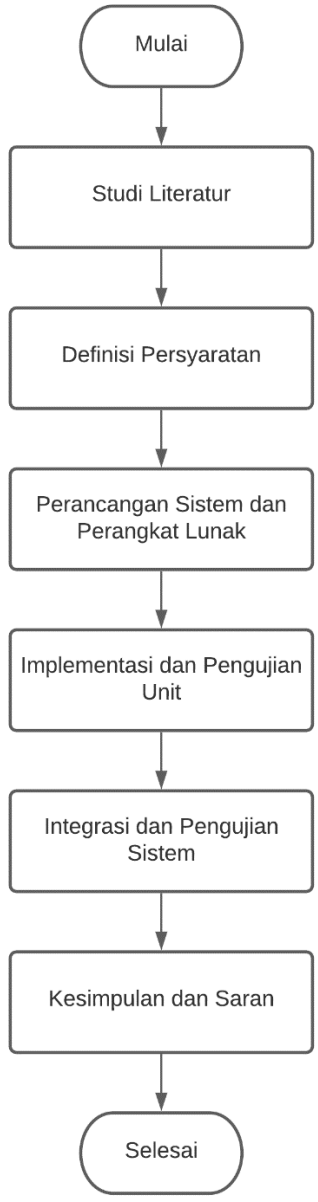
6. Hitung perubahan matriks partisi:

$$\mu_{ik} = \frac{\left[\sum_{j=1}^{p}\left(X_{kj} - V_{ij}\right)^2\right]^{\frac{-1}{p-1}}}{\sum_{i=1}^{c}\left[\sum_{j=1}^{p}\left(X_{kj} - V_{ij}\right)^2\right]^{\frac{-1}{p-1}}}$$

7. Cek kondisi berhenti:
   a. Jika Jika ($|P_t - P_{t-1}| < \xi$) atau (t < iterasi maksimal) maka berhenti;
   b. Jika tidak: maka $t = t + 1$ kemudian ulang langkah ke-4.

BAB 3 METODOLOGI

Bagian ketiga berisii dengan metodologi dengan topik bahasan mengenai langkah-langkah atau metode yang digunakan dalam pengembangan web *e-learning* codemaniac. Diagram alur metodologi ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alur Metode Penelitian

3.1 Studi Literatur

Prosedur riset harus dilengkapi dengan tahapan studi literatur yang digunakan sebagai rujukan untuk menyusun penelitian ini serta pengembangan

untuk sistem itu sendiri. Beberapa dasar teori yang dapat bermanfaat dalam pengembangan antara lain:

1. Kajian Pustaka
2. *E-Learning*
3. Perilaku Pengguna
4. Web Adaptif
5. *Waterfall Software Development Life Cycle*
6. *Unified Modeling Language* (UML)
7. *Data Flow Diagram* (DFD)
8. Pengujian *Black Box*
9. Pengujian *White Box*
10. *Log File*
11. Algoritma *Fuzzy Cmeans*

Sumber yang digunakan bisa didapatkan dari buku, jurnal, laporan ilmiah, serta bantuan *search engine* yang ada diinternet untuk memperkuat dasar teori terkait dengan pengembangan web *e-learning* codemaniac.

3.2 Definisi Persyaratan

Pada tahap analisis kebutuhan dilakukan penggalian kebutuhan terhadap *stakeholder* tentang daftar kebutuhan yang hendak dikembangkan kemudian memodelkannya dan membuat *Use Case Diagram* dari kebutuhan yang telah didapatkan. Tahap analisis kebutuhan ini dapat dilakukan lebih dari satu kali apabila belum sesuai dengan tujuan dari *stakeholder*. Pengulangan dilakukan dengan tujuan untuk memperbaiki sistem tersebut agar seseuai dengan tujuan yang diharapkan.

3.3 Perancangan Sistem

Tahap perancangan sistem, sistem dirancang berdasarkan hasil yang didapatkan pada fase analisis kebutuhan. Tahapan ini mendefinisikan kebutuhan ke dalam model perancangan menggunakan *Unified Modelling Language* (UML) untuk pendekatan berorientasi objek. Hasil dari perancangan berorientasi objek adalah *Sequence Diagram* dan *Class Diagram* untuk kemudian dilanjutkan dengan perancangan *user interface*. Seluruh hasil perancangan akan digunakan sebagai rujukan untuk kemudian diimplementasikan. Pendekatan prosedural juga digunakan dalam penelitian ini untuk menghasilkan *data flow diagram.*

3.4 Implementasi

Implementasi sistem dilakukan dengan menggunakan pendekatan berorientasi objek dan prosedural. Web *e-Learning* Codemaniac ini akan memiliki

fitur tambahan sesuai dengan yang telah didapatkan pada proses definisi persyaratan.

3.5 Integrasi dan Pengujian Sistem

Bagian ini diisi dengan langkah pengujian sistem secara keseluruhan, untuk memastikan bahwa sistem yang telah diimplementasikan dapat beroperasi sesuai dengan rancangan dan kebutuhan. Proses pengujian dilaksanakan dengan teknik *white box testing* dan *black box testing*. Program yang telah dikembangkan akan diuji untuk memastikan semua unit sistem bekerja dengan tepat dalam satu kesatuan serta sistem yang dihasilkan sesuai dengan analisis kebutuhan yang telah dilakukan diawal. Apabila ditemukan sebuah kekurangan atau ada akan ada perubahan fungsionalitas atau fitur, maka dapat dicatat untuk diperbaiki pada pengembangan berikutnya.

3.6 Kesimpulan dan Saran

Penarikan kesimpulan akan dihasilkan setelah seluruh proses pada metode *waterfall* dilaksanakan. Kesimpulan akhir dapat diambil berdasarkan pengembangan sistem, dimulai dengan analisis kebutuhan hingga pengujian sistem yang telah dilalui. Pemberian saran juga akan berguna untuk rekomendasi perbaikan kesalahan yang terjadi serta dapat digunakan sebagai tolak ukur bagi penelitian selanjutnya.

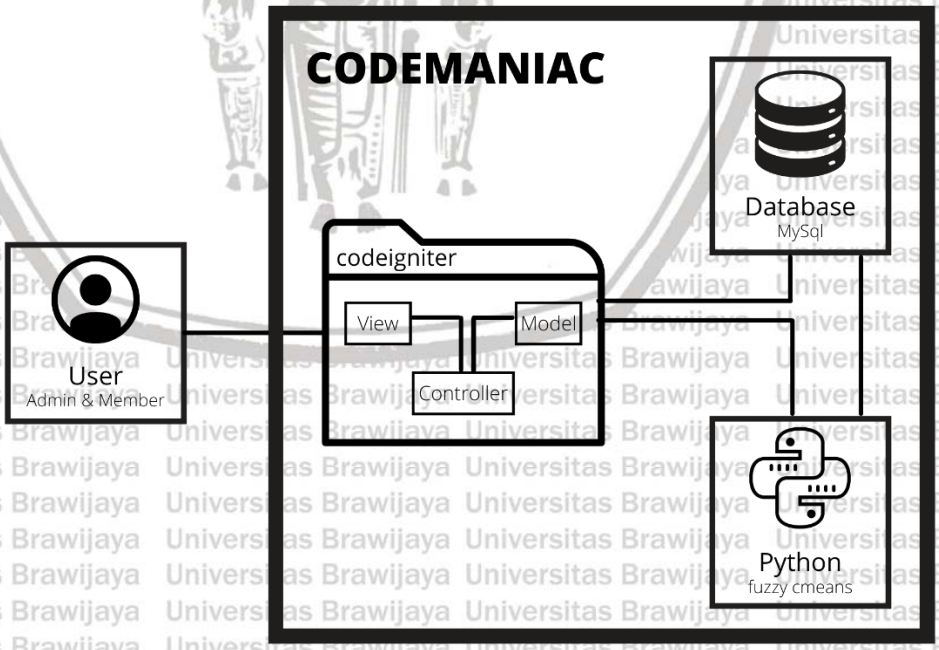
BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran Umum Sistem

Penelitian pengembangan lanjut web codemaniac menjadi lebih adaptif berdasarkan perilaku pengguna yang tersimpan dalam *log* sistem akan digambarkan secara umum dalam dua bagian, yaitu deskripsi umum sistem dan lingkungan sistem.

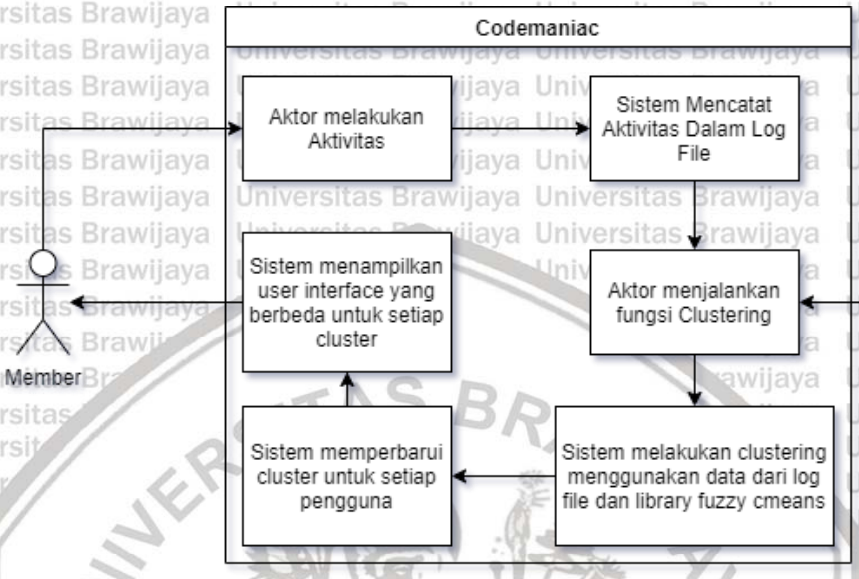
4.1.1 Deskripsi Umum Sistem

Codemaniac merupakan suatu sarana pembelajaran yang atraktif untuk pemrograman *java* dengan basis *online* yang disusun untuk meningkatkan daya tarik dan pengetahuan mahasiswa FILKOM UB terhadap pemrograman *java* (Bastari, et al., 2017). Dalam sistem ini terdapat berbagai latihan soal dari beberapa materi, selain itu terdapat fitur untuk memberikan tantangan antar pengguna. Sistem ini juga menggunakan teknik *gamification* sehingga dapat menyajikan sistem yang atraktif, salah satunya dengan adanya poin pengalaman, poin aktivitas, dan *badge* sesuai dengan aktivitas pengguna tersebut. Penelitian yang dilakukan saat ini akan menjadikan Codemaniac lebih adaptif dengan cara menggunakan rekaman perilaku setiap pengguna yang ada pada *log file* dalam *database* untuk kemudian dibagi dalam 3 *cluster* menggunakan algoritma *fuzzy cmeans* dengan memanfaatkan bahasa pemrograman *python*, setiap *cluster* akan memiliki antarmuka yang berbeda sesuai dengan pengaturan pada sistem. Untuk visualisasi rancangan sistem dapat dilihat pada Gambar 4.1.



Gambar 4.1 Arsitektur Sistem Codemaniac

Kerangka penelitian pada Gambar 4.2 menunujukkan cara sistem menerima interaksi dari aktor member dan admin. Gambar 4.2 menunjukkan alur sistem untuk melakukan clustering untuk member dengan merekam interaksi member untuk disimpan dalam log file, kemudian seorang admin akan mentrigger sistem untuk melakukan fungsi clustering sehingga cluster dari semua member akan diperbaharui.



Gambar 4.2 Kerangka Penelitian

4.1.2 Lingkungan Sistem

Codemaniac telah dibangun dengan bahasa pemrograman PHP pada *framework* CodeIgniter versi 3.1.3, kemudian menggunakan MySQL sebagai basis datanya, serta menggunakan *framework* MaterializeCSS untuk antarmuka halaman pengguna dan *framework* Bootstrap untuk antarmuka halaman admin. Dalam pengembangan kali ini akan ditambahkan bahasa pemrograman *python* versi 3.9 untuk menjalankan fungsi *clustering*.

4.2 Identifikasi Aktor

Pada Tabel 4.1 terdapat penjelasan seputar peran dari aktor yang terkait dengan sistem. Aktor tersebut diperoleh pada langkah analisis kebutuhan terdiri dari Admin, *Guest*, dan Member. Guna mengidentifikasi aktor terkait, Tabel 4.1 berisi penjelasan yang dibutuhkan seputar aktor.

Tabel 4.1 Identikasi Aktor

No.	Nama Aktor	Deskripsi
1.	Admin	Admin adalah aktor yang dapat melihat *dashboard* halaman admin, melihat *Log user*, melihat daftar *cluster*, serta melakukan *clustering*.
2.	*Guest*	*Guest* adalah aktor yang akan menggunakan sistem, *Guest* hanya berada pada halaman *login*

		atau *register*.
3.	Member	Member adalah aktor yang telah berhasil melakukan *login* ke dalam sistem, Member dapat menerima perubahan pada tampilan antarmuka pengguna.

4.3 Daftar kebutuhan fungsional

4.3.1 Aturan Penomoran

Berdasarkan pada Gambar 4.3 dan Tabel 4.2, aturan penomoran yang diperlukan dalam spesifikasi kebutuhan untuk peneletian ini.

KODE : COMA - X - 000

Gambar 4.3 Aturan Penomoran Kebutuhan

Tabel 4.2 Penjelasan Aturan Penomoran Kebutuhan

Aturan Penomoran	Deskripsi
Coma	Merupakan kode yang berisi nama sistem.
X	Merupakan kode representasi kebutuhan dengan pengodean 'F' untuk Kebutuhan Fungsional dan 'NF' untuk Kebutuhan Non-Fungsional.
000	Merupakan nomor urut representasi kebutuhan.

4.3.2 Kebutuhan Fungsional

Kebutuhan yang didapatkan telah dituliskan pada Tabel 4.3 namun masih bersifat umum dan belum dispesifikasikan. Dalam penelitian ini terdapat lima kebutuhan yang harus terpenuhi.

Tabel 4.3 Kebutuhan Fungsional

No.	Kebutuhan
1.	Sistem dapat menampilkan daftar member beserta *cluster*-nya
2.	Sistem dapat melakukan *clustering* berdasarkan data pada tabel *log*
3.	Sistem dapat menampilkan *dashboard* pada halaman admin
4.	Sistem dapat merubah antarmuka pengguna sesuai dengan *cluster* pengguna tersebut

5.	Sistem dapat memeriksa *cluster* pengguna saat pengguna *login*

Tabel 4.4 Kebutuhan Non - Fungsional

No.	Kebutuhan
1.	*Compability*

4.3.3 Spesifikasi Kebutuhan Fungsional

Setelah membuat daftar kebutuhan fungsional, kemudian kebutuhan tersebut dispesifikasikan dan mendapatkan kode penamaan sesuai dengan aturan pada pembahasan sebelumnya. Sesuai dengan hasil analisis kebutuhan, telah dikumpulkan sebanyak lima kebutuhan fungsional dilkengkapi dengan spesifikasi dari setiap kebutuhan tersebut. Untuk daftar kebutuhan fungsional beserta spesifikasinya akan dijelaskan pada Tabel 4.5. dan untuk tabel kebuthan non-fungsional dapat dilihat pada Tabel 4.6.

Tabel 4.5 Spesifikasi Kebutuhan Fungsional OOP

No.	Kode	Nama Fungsi	Deskripsi
1.	COMA-F-01	Melihat daftar *cluster*	Sistem dapat menampilkan hasil *clustering* beserta member dalam setiap *cluster*.
2.	COMA-F-02	Melakukan *clustering*	Sistem dapat melakukan *clustering* dari tabel *log* yang terdapat dalam database codemaniac.
3.	COMA-F-03	Menampilkan *dashboard*	Sistem dapat menampilkan *dashboard* pada halaman admin, yang berisi total member, total *question*, total categori, total *exercise* yang telah dikerjakan, member dengan pengalaman bermain tertinggi, member dengan aktifitas terbanyak.
4.	COMA-F-04	*Login*	Pembaruan pada fungsi *login* yang sudah ada, dengan menambahkan pemeriksaan *cluster*.
5.	COMA-F-05	Perubahan antarmuka pengguna	Sistem dapat mengubah antarmuka pengguna sesuai dengan *cluster* yang pengguna tersebut.

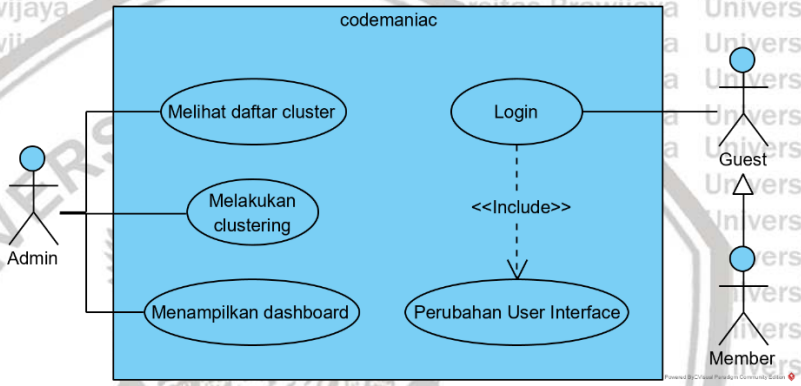
4.4 Kebutuhan Non-Fungsional

Tabel 4.6 Kebutuhan Non-Fungsional

No.	Kode	Nama fungsi	Deskripsi
1.	COMA-NF-1	*Compability*	Peranda seperti *Microsoft Edge, Mozilla Firefox, Google Chrome* dan *Safari* dapat mengakses sistem.

4.5 Pemodelan Kebutuhan

4.5.1 *Use Case Diagram*



Gambar 4.4 Use Case Diagram

Dapat dilihat pada Gambar 4.4 terdapat aktor admin yang dapat melakukan fungsi melihat daftar *cluster*, melakukan *clustering*, dan menampilkan *dashboard* serta aktor *Guest* yang dapat melakukan *Login* dan menerima perubahan *User Interface* saat menjadi Member.

4.5.2 Use Case Scenario

Sesuai *use case diagram* pada Gambar 4.4, maka sistem Codemaniac memiliki *use case scenario* yang dapat dilihat pada Tabel 4.7 – Tabel 4.11.

Tabel 4.7 *Use case Scenario* Melihat daftar *cluster*

Use Case Scenario Melihat daftar *cluster*	
Tujuan	Melihat daftar *cluster* beserta nama member dalam *cluster* tersebut
Aktor	Admin
Kondisi awal	Aktor telah berhasil masuk kedalam halaman admin
Main flow	1. Aktor memilih menu *content management* 2. Sistem menampilkan daftar menu tersebut

	3. Aktor memilih opsi *cluster* 4. Sistem menampilkan daftar *cluster*
Scenario alternatif	-
Kondisi akhir	Aktor dapat melihat daftar *cluster* beserta member dalam *cluster* tersebut

Tabel 4.8 *Use Case Scenario* Melakukan *Clustering*

Use Case Scenario Melakukan *Clustering*	
Tujuan	Melakukan *clustering* berdasarkan tabel *log* pada *database*
Aktor	Admin
Kondisi awal	Aktor telah berada pada halaman *cluster*
Main flow	1. Aktor menekan tombol start *clustering* 2. Sistem memperbarui daftar *cluster*
Scenario alternatif	-
Kondisi akhir	Daftar *cluster* akan menampilkan daftar terbaru dari hasil *clustering* terkini

Tabel 4.9 Use Case Scenario Menampilkan Dashboard

Use Case Scenario Menampilkan Dashboard	
Tujuan	Melihat dashboard halaman admin yang berisi beberapa informasi
Aktor	Admin
Kondisi awal	Aktor telah berhasil masuk pada halaman admin
Main flow	1. Aktor menekan menu Dashboard 2. Sistem menampilkan halaman dashboard
Scenario alternatif	-
Kondisi akhir	Aktor dapat melihat dashboard halaman admin yang berisi beberapa informasi

Tabel 4.10 *Use Case Scenario Login*

Use Case Scenario Login	
Tujuan	Aktor berhasil masuk kedalam sistem
Aktor	*Guest*
Kondisi awal	*Guest* telah berada pada halaman *Login*
Main flow	1. *Guest* mengisi *username* 2. *Guest* mengisi *password* 3. *Guest* menekan tombol *sign in* 4. Sistem memeriksa *username* dan *password Guest* 5. Sistem memeriksa *cluster* Aktor
Scenario alternatif	1. *Username* dan *password* tidak ada pada *database* atau tidak sesuai dengan *database* 2. Menampilkan kembali halaman *login*
Kondisi akhir	Member berhasil masuk pada sistem

Tabel 4.11 *Use Case Scenario* Perubahan Antarmuka Pengguna

Use Case Scenario Perubahan Antarmuka Pengguna	
Tujuan	Sistem merubah *user interface* pengguna sesuai dengan *cluster* pengguna tersebut
Aktor	Member
Kondisi awal	Member telah berhasil *Login*
Main flow	1. *User interface* bertema merah untuk cluster 1, bertema biru untuk *cluster* 2, dan bertema hijau untuk *cluster* 3 atau belum memiliki *cluster*
Scenario alternatif	-
Kondisi akhir	*User interface* member berubah sesuai dengan *cluster* member

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan Sistem

Tahap perancangan sistem memiliki relasi dengan proses analisis kebutuhan, dengan menggunakan keluaran dari proses analisis kebutuhan sebagai dasar pada proses perancangan. Tahap perancangan sistem codemaniac akan terbagi ke dalam beberapa bagian yaitu Perancangan arsitektur, Perancangan komponen, Perancangan data, dan Perancangan antarmuka.

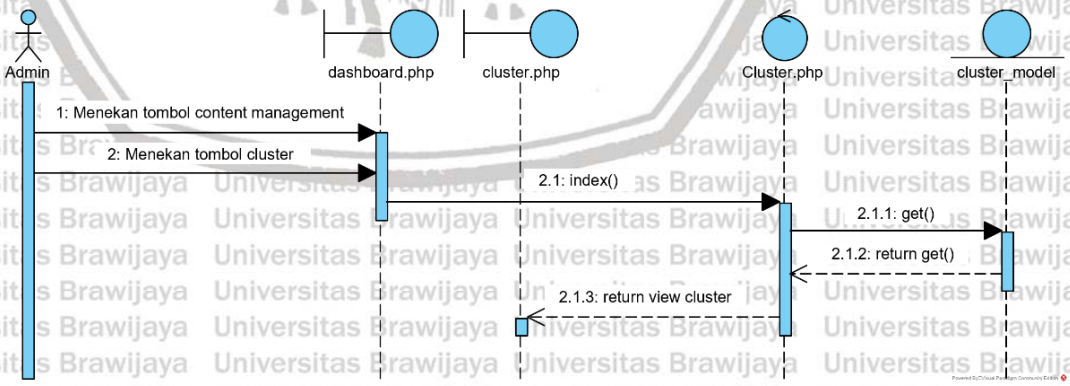
5.1.1 Perancangan Arsitektur

Perancangan arsitektur sistem ini menggunakan dua pendekatan yaitu, pendekatan berorientasi objek dan pendekatan prosedural yang dapat dilihat pada Tabel 5.1. Tahap perancangan arsitektur dengan pendekatan berorientasi objek, berisi dengan interaksi antara suatu *class* dengan *class* yang lainnya, kemudian juga terdapat alur dari transfer data antar objek yang digambrakan dengan *sequence diagram* dan juga mendefinisikan relasi antar *class* dalam sistem yang ditampilkan dengan *class diagram*. Sedangkan untuk tahapan dengan pendekatan prosedural membahas proses *clustering* yang akan direpresentasikan dalam *Data Flow Diagram*.

Tabel 5.1 Metode Perancangan Arsitektur Sistem

No.	Metode	Nama Diagram
1.	Berorientasi Objek	*Sequence Diagram*
2.		*Class Diagram*
3.	Prosedural	*Data Flow Diagram*

5.1.1.1 *Sequence Diagram* Melihat Daftar Cluster

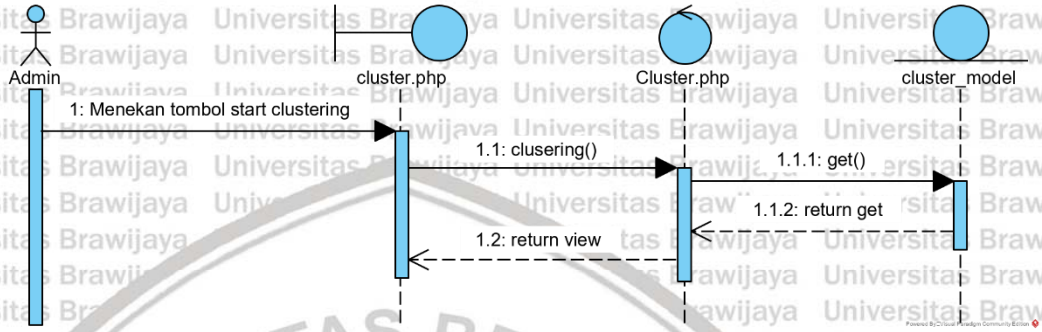


Gambar 5.1 *Sequence Diagram* Melihat Daftar *Cluster*

Pada Gambar 5.1 ditunjukkan *sequence diagram* melihat daftar *cluster*. *Sequence diagram* ini memiliki empat macam notasi yang tersusun dari seorang aktor yaitu admin, dua buah *boundary* yaitu dashboard.php dan cluster.php, sebuah *controller* yaitu Cluster.php, dan sebuah *entity* yaitu cluster_model.

Dalam menjalankan *use case* melihat daftar *cluster*, aktor berada pada halaman *dashboard* untuk kemudian aktor menekan menu *content management* kemudian menekan menu *cluster* yang akan memanggil method index pada *controller* Cluster.php. *Controller* index tersebut akan mengambil data dari entitas cluster_model dan memberikan respon untuk disimpan pada *controller* index dan ditampilkan pada halaman cluster.php.

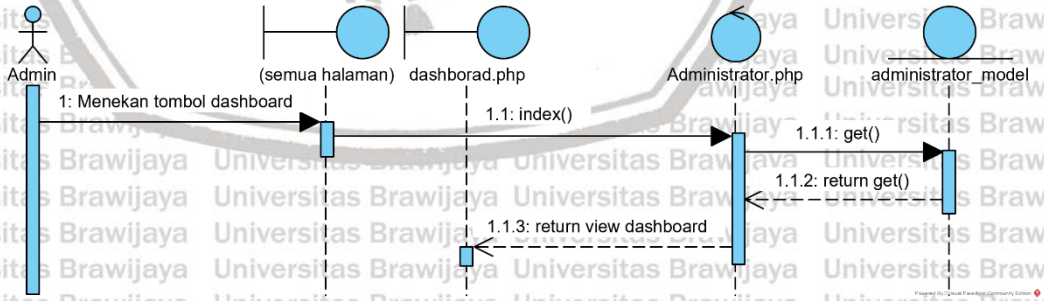
5.1.1.2 *Sequence Diagram* Melakukan *Clustering*



Gambar 5.2 *Sequence Diagram* Melakukan *Clustering*

Sequence diagram Melakukan *Clustering* dapat dilihat pada Gambar 5.2. dalam *sequence diagram* tersebut, terdapat seorang aktor yaitu Admin, sebuah *boundary* yaitu cluster.php, sebuah *controller* yaitu, Cluster.php dan sebuah *entitas* yaitu cluster_model. Proses melakukan *clustering* diawali dengan admin yang telah berada pada halaman cluster.php, admin kemudian menekan tombol *start clustering* yang memanggil *method* clustering pada *controller* Cluster.php. *method* clustering akan menjalankan proses *clustering* menggunakan *fuzzy cmeans* kemudian meminta data pada cluster_model dan cluster_model memberikan *response* yang disimpan dalam *controller* Cluster.php untuk ditampilkan pada halaman cluster.php.

5.1.1.3 *Sequence Diagram* Menampilkan *Dashboard*

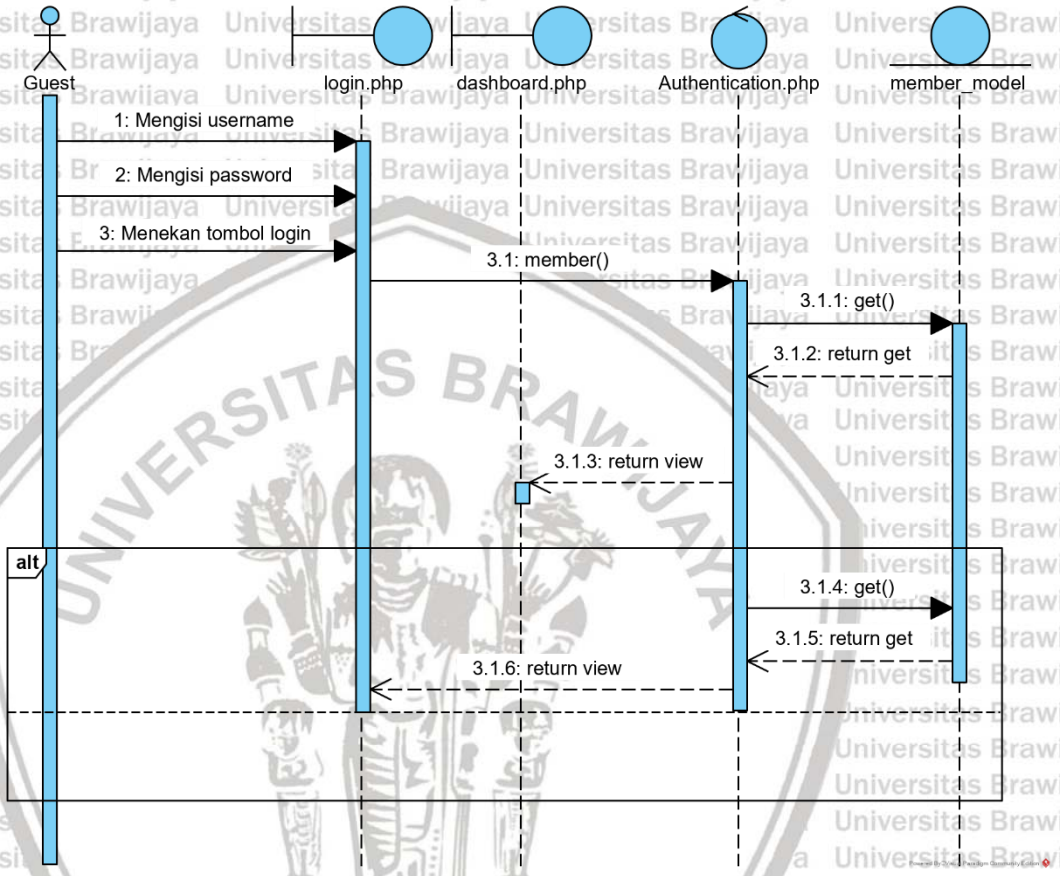


Gambar 5.3 Sequence Diagram Menampilkan *Dashboard*

Pada Gambar 5.3 terdapat *sequence diagram* menampilkan *dashboard*, di dalam *sequence diagram* tersebut terdapat seorang aktor yaitu Admin, kemudian memiliki dua buah *boundary* yang pertama bisa berisi seluruh halaman yang dapat diakses seorang Admin setelah berhasil *Login* dan halaman dashboard.php, kemudian memiliki sebuah *controller* yaitu Administrator.php, dan sebuah entitas bernama administrator_model. *Use case* akan berjalan saat

aktor menekan menu *dashboard* pada halaman yang sedang ditampilkan, kemudian sistem akan mengarahkan menuju *method* index pada Administrartor.php untuk meminta data dari administrator_model yang akan menyimpan *response* pada *controller* Administrator.php untuk ditampilkan pada halaman dashboard.php.

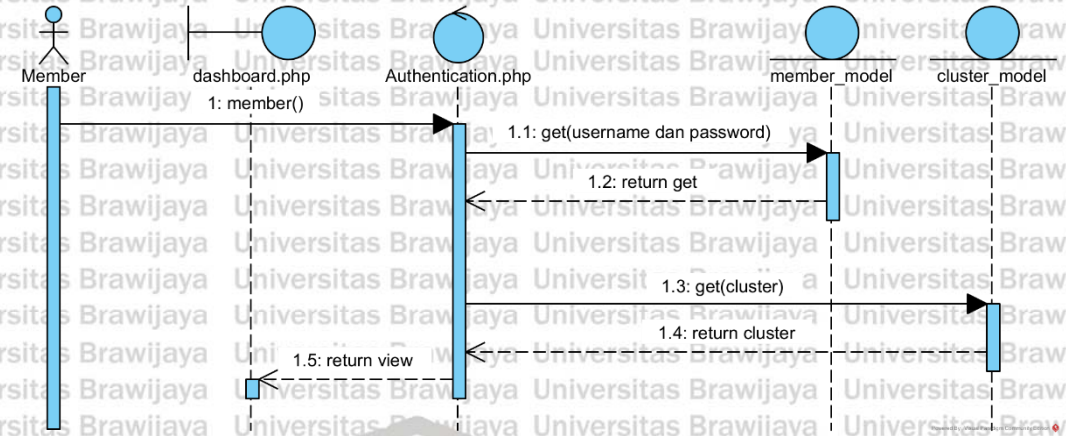
5.1.1.4 *Sequence Diagram Login*



Gambar 5.4 *Sequence Diagram Login*

Pada Gambar 5.4 terdapat *sequence diagram* melakukan *Login*. *Sequence diagram Login* memiliki *mainflow* dan sebuah *alternatif flow*. *Mainflow sequence diagram Login* dimulai dengan aktor *Guest* yang mengisi *username* dan *password* kemudian menekan tombol *login* yang memanggil *method* member pada *controller* Authentication.php untuk mengecek *username, password* dan *cluster* dari aktor *Guest* yang disimpan kembali dalam *controller* Authentication.php, jika berhasil maka sistem akan mengarahkan aktor kembali pada halaman dashboard.php. *alternatif flow* akan berjalan saat aktor *Guest* salah dalam memasukkan *username* atau *password*, dan akan diarahkan kembali pada halaman login.php

5.1.1.5 *Sequence Diagram* Perubahan Antarmuka Pengguna

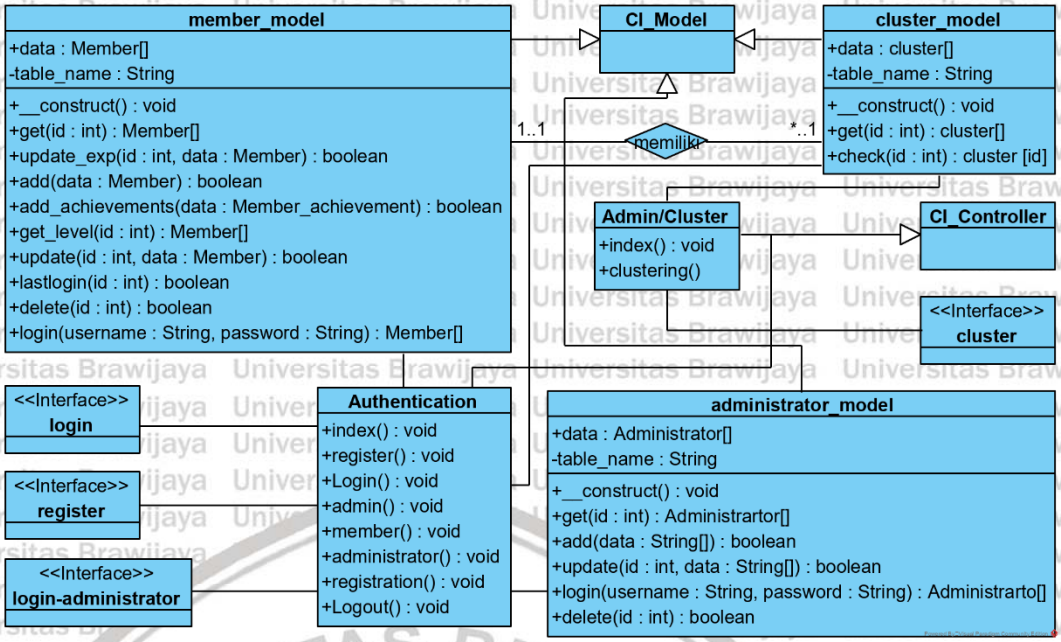


Gambar 5.5 *Sequence Diagram* Perubahan Antarmuka Pengguna

Pada Gambar 5.5 ditunjukkan *sequence diagram* perubahan antarmuka pengguna. *Sequence diagram* ini memiliki seorang aktor yaitu member, kemudian memilki sebuah *controller*, yaitu Authentication.php serta memiliki dua buah entitas yaitu member_model dan cluster_model. Perubahan antarmuka pengguna berjalan saat user melakukan login dengan memanggil *method* member pada *controller* Authentication.php, *controller* tersebut meminta *username* dan *password* dari member_model dan dikembalikan pada *controller* kemudian meminta *cluster* dari *username* yang terkait pada cluster_model. Setelah mendapatkan *username*, *password* dan *cluster* controller akan mengarahkan sistem pada halaman dashboard.php.

5.1.1.6 Class Diagram

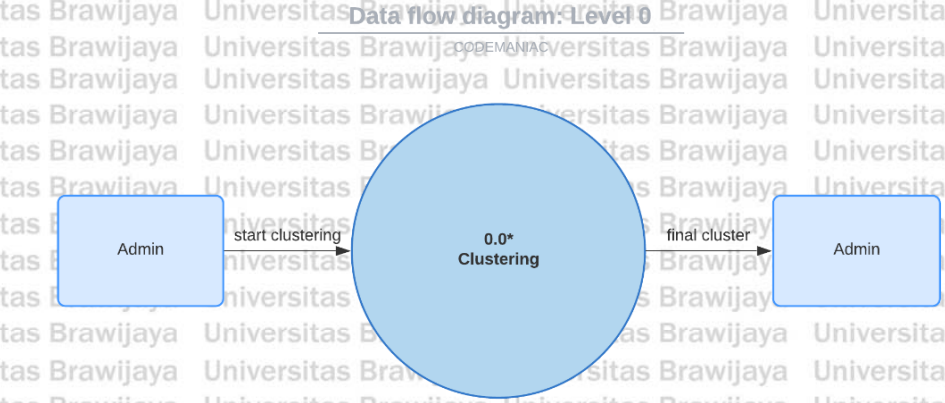
Pada Gambar 5.6 terdapat *class diagram* dari pengembangan lanjut sistem *e-learning* codemaniac. *Class diagram* ini bertujuan untuk menjelaskan susunan *class-class* serta relasi yang terkait antar *class* tersebut. Dalam *framework* CodeIgniter terdapat tiga jenis *class* yaitu, *class controller* untuk menampung semua *controller* pada sistem, *class model* untuk menampung semua model pada sistem, dan *class view* untuk menampung semua viiew pada sistem. Pada *class controller* terdapat sedikit pembaruan pada *controller* Authentication, serta penambahan *controller* baru yaitu Admin/Cluster. Untuk *class model* terdapat pembaruan relasi dari member_model ke *class model* baru yaitu cluster_model, pembaruan relasi *one-to-one* dari member_model ke cluster_model dan *one-to-many* dari cluster_model kepada member_cluster. Pada *class view* juga terdapat penambahan halaman *cluster* untuk menampilkan daftar *cluster*.



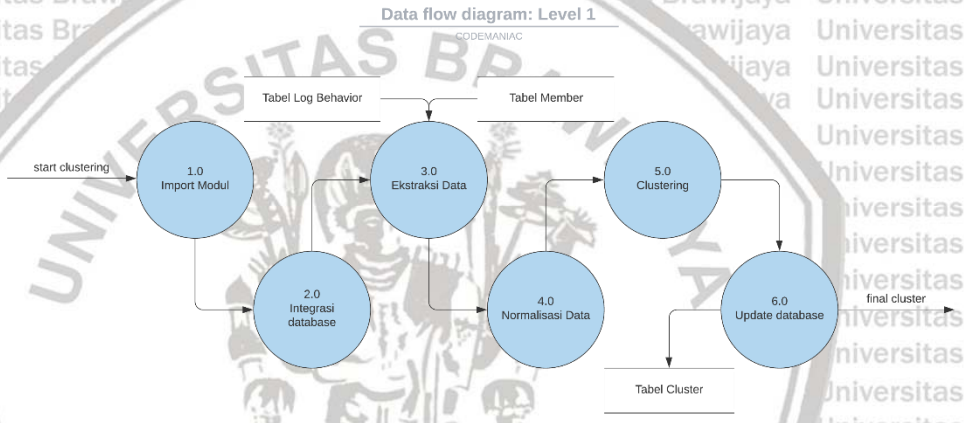
Gambar 5.6 *Class Diagram* Pengembangan Codemaniac

5.1.1.7 Data Flow Diagram Clustering

Pada Gambar 5.7 terdapat *Data Flow Diagram* (DFD) *Clustering* level 0. Dalam diagram tersebut aktor admin melakukan *start clustering* yang direspon sistem dengan menjalankan fungsi *clustering* sehingga menghasilkan *cluster* untuk ditampilkan kembali kepada admin. Kemudian pada Gambar 5.8 terdapat DFD Clustering Level 1 yang merupakan leveling dari DFD *Clustering* Level 0, pada diagram tersebut setelah sistem menerima input perintah start *clustering*, sistem akan mengimpor modul-modul yang diperlukan. Selanjutnya sistem akan membuat koneksi dengan *database* sehingga dapat melakukan ekstraksi data untuk menggabungkan data dari tabel *log behavior* dan tabel member yang terdapat pada *database*. Data yang telah disusun kemudian dinormalisasi sebelum data diolah menggunakan *Fuzzy Cmeans* dalam Langkah ke 5.0. Data hasil *clustering* pada Langkah 5.0 kemudian dikirim Kembali ke *database* pada tabel *clustering* untuk kemudian ditampilkan kepada admin.



Gambar 5.7 *Data Flow Diagram Clustering* Level 0



Gambar 5.8 *Data Flow Diagram Clustering* Level 1

5.1.2 Perancangan Komponen

Bagian bab ini akan membahas secara mendetail dari sub-sistem setiap komponen untuk pengembangan sistem ini. Bab ini juga mendefinisikan tentang proses algoritma yang terjadi dengan format bahasa *pseudocode*. Sebagai pembahasan akan digunakan *method* member yang terdapat dalam *controller* Authentication, *method* clustering yang terdapat dalam *controller* Cluster, serta proses *clustering* yang terjadi pada *method* Clustering.

5.1.2.1 Perancangan Komponen Method Member

Pada tabel 5.1 terdapat algoritma *method* member yang berada pada *controller* Authentication. *Method* ini bertujuan untuk melakukan *login* oleh aktor. Algoritma ini akan menangkap inputan aktor berupa *username* dan *password*. *Username* dan *password* tersebut kemudian disimpan dalam satu variable untuk dilakukan pemeriksaan dengan data yang telah tersimpan pada *database*. Setiap menekan tombol *login*, sistem akan merekam tingkah tersebut dalam tabel *log*. Jika *username* atau *password* salah, maka akan mengarahkan

kembali pada halaman *login*. Jika *username* dan *password* benar maka data *lastlogin* pada member akan diperbaharui kemudian sistem menyimpan *session* yang berisi id *user*, *fullname user*, *username*, dan *cluster user* kemudian mengarahkan pada halaman profil.

Nama *class*: Authentication
Nama *method*: member

Tabel 5.2 *Pseudocode* Komponen *Method* Member

Pseudocode algoritma method member	
1	Begin
2	Deklarasi variable 'username' = input username;
3	Deklarasi variable 'password' = input password;
4	Deklarasi variable 'member' = menjalankan fungsi Login dari
5	model member dengan parameter (username, md5(password));
6	Deklarasi variable 'clusters' = menjalankan fungsi get dari
7	model cluster dengan parameter(member['id']);
8	Menjalankan fungsi add dari model log dengan parameter
9	(username, "Login");
10	if (!member) {
11	redirect('login');
12	} Else {
13	Menjalankan fungsi Lastlogin dari model member dengan
14	parameter (member['id']);
15	Menyimpan session 'uid' = member['id'];
16	Menyimpan session 'fullname' = member['fullname'];
17	Menyimpan session 'username' = member['username'];
18	Menyimpan session 'clusters' = clusters['clusters'];
19	Redirect('profile');
20	}
21	end

5.1.2.2 Perancangan Komponen Method Clustering

Pada Tabel 5.2 terdapat algoritma *method* clustering yang berada pada *controller* cluster. Tujuan dari *method* ini adalah untuk melakukan fungsi *clustering*. Algoritma diawali dengan deklarasi variable *command* yang menyimpan perintah untuk menjalankan *python* dan direktori *script* yang akan dieksekusi. Kemudian variable selanjutnya bernama *output* untuk menjalankan *variable command*. Selanjutnya terdapat *variable* data yang berbentuk *array* untuk menyimpan data dari fungsi *get*. Setelah itu menampilkan halaman mulai dari *header*, *cluster-list*, dan *footer*.

Nama *class*: Cluster

Nama *method*: Clustering

Tabel 5.3 *Pseudocode* Komponen *Method Clustering*

Pseudocode algoritma method clustering	
1	Begin
2	Deklarasi variable command = berisi fungsi escapeshellcmd
3	untuk menjalankan python serta direktori script clustering;
4	Deklarasi variable output = menjalankan fungsi shell_exac
5	dengan parameter command;
6	deklarasi variable 'data' = menjalankan fungsi get dari model
7	cluster;
8	mengarahkan kehalaman view 'admin/header';
9	mengarahkan kehalaman view 'admin-cluster-list', dan
10	menampilkan data;
11	mengarahkan kehalaman view 'admin-footer';
12	end

5.1.2.3 Perancangan Komponen Script Clustering

Pada Tabel 5.3 terdapat Algoritma proses *Clustering* yang bertujuan untuk mengelompokkan member menjadi tiga *cluster* sesuai dengan tingkah mereka yang telah tercatat pada tabel *log_behavior* pada *database*. Proses ini berjalan menggunakan bahasa pemrograman *python* yang terintegrasi dengan sistem pada *method clustering*. Proses *clustering* berawal dengan melakukan impor modul-modul yang diperlukan kemudian membuat koneksi dengan *database* untuk selanjutnya melakukan ekstraksi data dari *database* pada tabel member dan *log_behavior*. Data yang sudah terkumpul tadi digabungkan menjadi satu tabel dan dinormalisasi sebelum memulai proses *clustering* menggunakan modul *fuzzy cmeans*. Data yang telah di-*clustering*, kemudian diberi tanggal dan dikirim kembali ke *database* pada tabel *Clusters*.

Tabel 5.4 *Pseudocode* Komponen *Script Clustering*

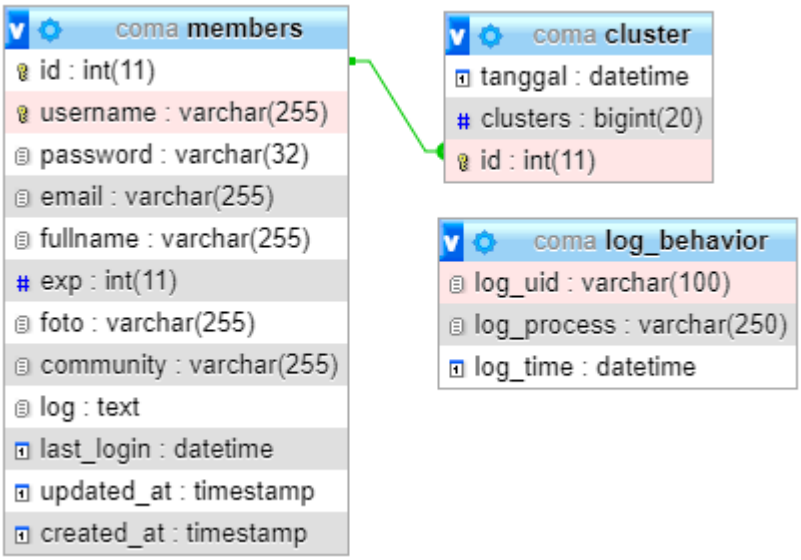
Pseudocode algoritma script clustering	
1	Begin
2	Impor modul mysql connector;
3	Impor modul sys;
4	Impor modul csv;
5	Impor modul pandas dan simpan sebagai = pd;
6	Impor modul numpy dan simpan sebagai = np;
7	Impor modul datetime dan simpan sebagai = dt;
8	Impor modul FCM;

```
Impor modul MinMaxScaler;
Impor modul pyplot dan simpan sebagai = plt;
Impor modul create_engine;
Deklarasi variable 'conn' untuk menyimpan koneksi dengan
database terkait;
Deklarasi variable 'mycursor' = conn.cursor;
Deklarasi variable 'engine' = untuk mengirim hasil clustering
ke database;
Mengarahkan mycursor pada id dan username dari tabel members;
Deklarasi variable df_members untuk menyimpan data member;
Mengarahkan mycursor pada log_uid dan log_process dari tabel
log;
Deklarasi variable df_log_behavior untuk menyimpan data log;
Deklarasi variable df_fusion untuk menggabungkan df_members
dan df_log_behavior berdasarkan kolom 'username';
Menghapus kolom 'username' dari df_fusion;
Deklarasi variable 'x' berisi integer 1-19;
Deklarasi variable 'count' = 0;
Deklarasi variable 'xl' = Panjang x;
Menambah 18 kolom pada tabel df_fusion;
Mengubah value kolom 'log process' menjadi angka 1;
Menghapus kolom 'log_process';
Deklarasi variable 'df_final' untuk mengumpulkan df_fusion
sesuai dengan kolom 'id';
Deklarasi variable 'df_floated' = df_final astype(float);
Deklarasi variable 'data_cluster' = df_floated iloc [:, :];
Deklarasi variable 'dataset' = np array(data_cluster);
Deklarasi variable 'data' = dataset tolist;
Deklarasi variable 'scaler' = MinMaxScaler;
Deklarasi variable 'data_minmax' = np around;
Deklarasi variable 'fcm' = FCM dengan 3 cluster;
Set fcm.fit(data_minmax);
Deklarasi variable 'fcm_centers' = fcm centers;
Deklarasi variable 'fcm_labels' = fcm u argmax;
Set df_final kolom clusters = fcm_label;
Insert tanggal pada df_final;
Deklarasi variable 'df_temp' = df_final[tanggal,cluster,id];
Send df_temp to sql;
```

```
47  Set mycursor primary key 'id';
48  Set mycursor modify column id = int(11);
49  Set mycursor foreign key id cluster pada id members;
50  Set mycursor create index pada cluster;
51  end
```

5.1.3 Perancangan Basis Data

Pada Gambar 5.9 terdapat rancangan data dalam bentuk *Physical Data Model* (PDM) dari beberapa tabel yang terkait dengan penelitian. Pada pengembangan sistem codemaniac ini terdapat penambahan tabel *cluster*.



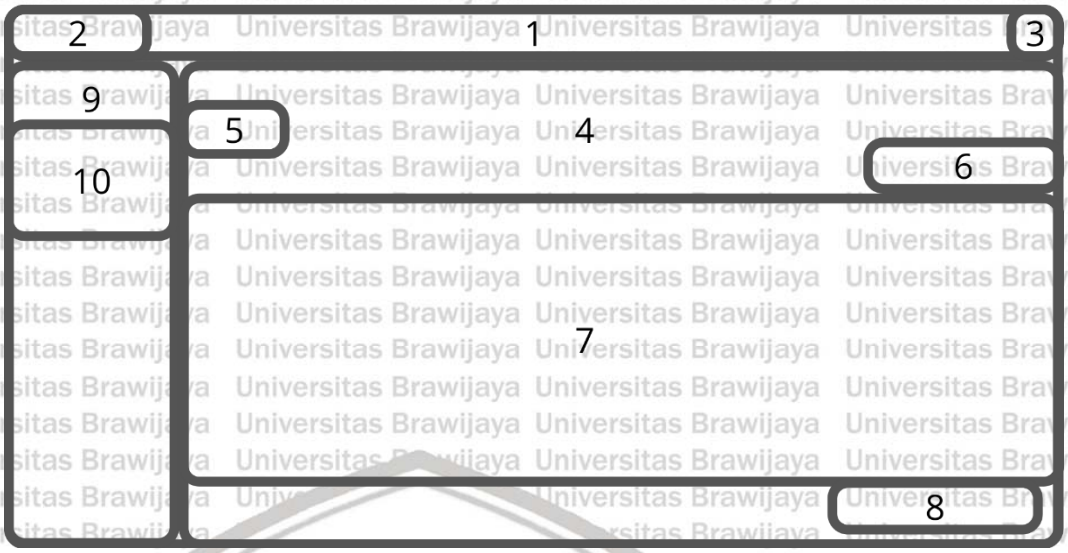
Gambar 5.9 Perancangan *Physical Data Model*

5.1.4 Perancangan Antarmuka

Bagian sub-bab perancangan antarmuka memapaparkan seputar rancangan tampilan dari pengembangan sistem codemaniac. Rancangan ini akan menunjukkan komponen dalam halaman-halaman yang mengalami perubahan. Hasil rancangan antarmuka ini akan menjadi media untuk antara interaksi sistem dengan pengguna. Perancangan antarmuka yang akan disusun adalah perancangan antarmuka *List Cluster* dan perancangan antarmuka perubahan *interface* yang tersaji pada sub-bab 5.1.4.1 – 5.1.4.3.

5.1.4.1 Perancangan Antarmuka List Cluster

Perancangan antarmuka *List Cluster* dapat diperhatikan pada Gambar 5.10. terdapat sepuluh komponen penyusun halaman antarmuka tersebut. Keterangan untuk setiap komponen tersebut akan dijabarkan pada Tabel 5.4.



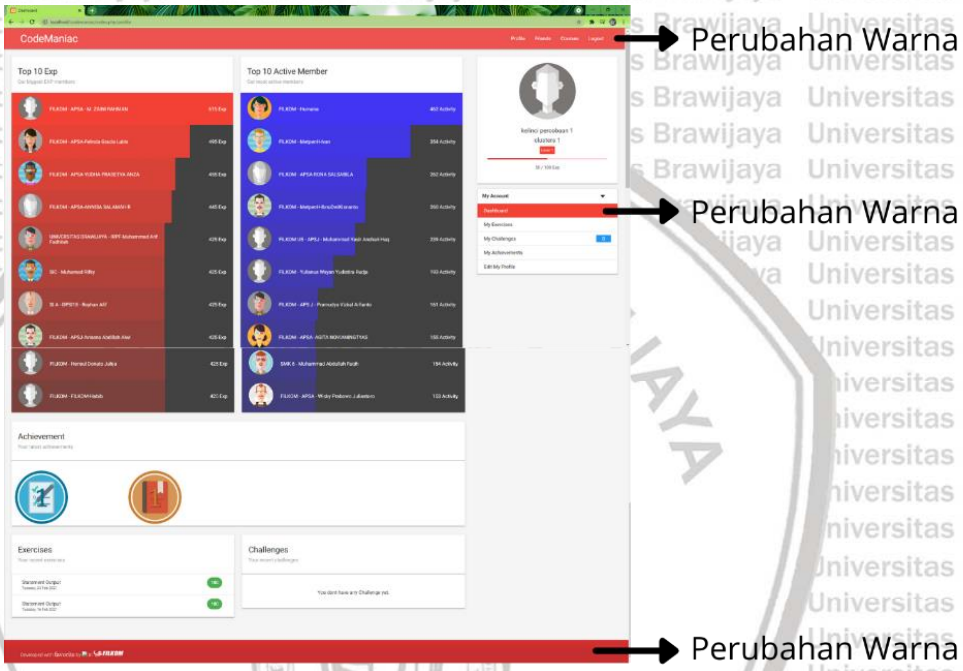
Gambar 5.10 Perancangan Antarmuka *List Cluster*

Tabel 5.5 Keterangan Gambar Perancangan Antarmuka *List Cluster*

No	Nama Objek	Tipe	Keterangan
1	*Header*	*Header*	Menampilkan *Header* dengan tema yang sama dengan halaman yang lainnya
2	Nama Sistem	Gambar	Menampilkan nama sistem codemaniac pada halaman administrator
3	*Log Out*	Tombol	Menampilkan *icon logout* sebagai akses keluar dari sistem
4	*Body*	*Body*	Menampilkan menu utama pada halaman *cluster*
5	*Start Clustering*	Tombol	Menampilkan tombol yang digunakan untuk memulai *clustering*
6	*Search*	Kolom	Menampilkan kolom pencarian untuk mempermudah memilah data
7	Tabel *Cluster*	Tabel	Menampilkan tabel hasil *clustering*
8	*Page information*	Tombol	Menampilkan tombol untuk berpindah halaman tabel *cluster*
9	*Side bar*	*Bar*	Menampilkan *sidebar* yang sama pada setiap halaman
10	*Menu bar*	*Dropdown*	Menampilkan menu yang terdapat pada halaman administrator

5.1.4.2 Perancangan Antarmuka Perubahan Interface

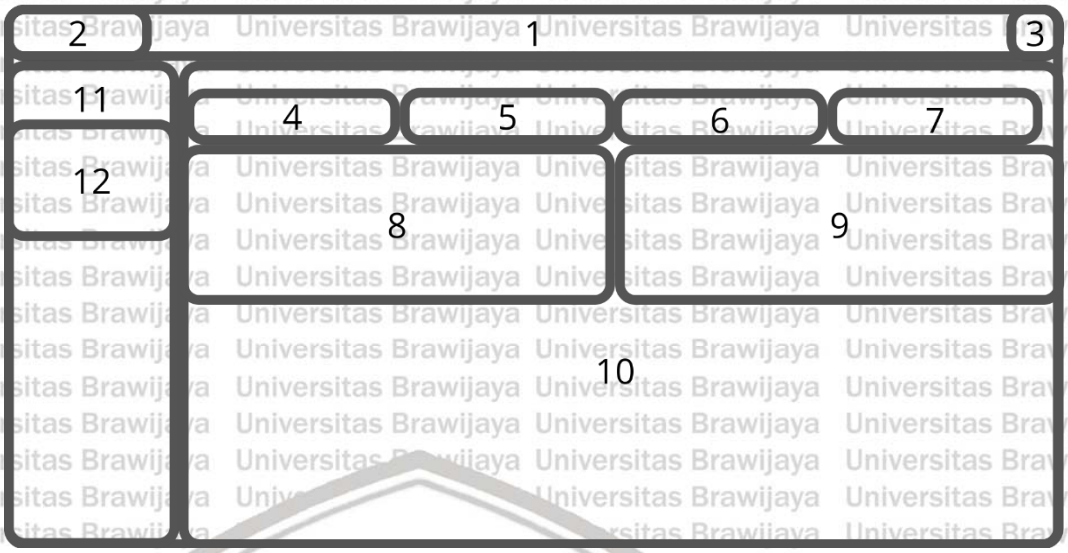
Perancangan antarmuka Perubahan Interface ditunjukkan pada Gambar 5.11. Terdapat tiga bagian yang mengalami perubahan warna yaitu ada pada *header*, *sidebar*, dan *footer*. Untuk member yang mendapatkan golongan *cluster* satu, maka warna yang akan muncul adalah warna merah. Untuk member yang tergolong *cluster* dua, maka warna yang akan muncul pada *interface* mereka adalah warna biru. Sedangkan untuk member yang tergolong *cluster* tiga atau belum mendapatkan *cluster*, maka warna pada *user interface* mereka adalah warna hijau. Perubahan terdapat pada semua halaman yang dapat diakses oleh Member.



Gambar 5.11 Perancangan Antarmuka Perubahan *User Interface*

5.1.4.3 Perancangan Antarmuka Dashboard

Perancangan antarmuka *Dashboard* dapat diperhatikan pada Gambar 5.12. Dalam perancangan ini terdapat dua belas komponen penyusun. Keterangan untuk setiap komponen tersebut akan dijabarkan pada Tabel 5.5.



Gambar 5.12 Perancangan Antarmuka *Dashboard*

Tabel 5.6 Keterangan Gambar Perancangan Antarmuka *Dashboard*

No	Nama Objek	Tipe	Keterangan
1	*Header*	*Header*	Menampilkan *Header* dengan tema yang sama dengan halaman yang lainnya
2	Nama Sistem	Gambar	Menampilkan nama sistem codemaniac pada halaman administrator
3	*Log Out*	Tombol	Menampilkan *icon logout* sebagai akses keluar dari sistem
4	Total Member	Panel	Menampilkan informasi total member
5	Total *Questions*	Panel	Menampilkan informasi total *questions*
6	Total *Category*	Panel	Menampilkan informasi total *category*
7	*Exercise Worked*	Panel	Menampilkan informasi *exercise* yang telah dikerjakan
8	Top 10 *Experience*	Panel	Menampilkan informasi sepuluh member dengan level tertinggi
9	Top 10 *Active*	Panel	Menampilkan informasi sepuluh member dengan keaktifan tertinggi
10	*Body*	*Body*	Menampilkan menu utama pada halaman *cluster*
11	*Sidebar*	*Bar*	Menampilkan *sidebar* yang sama pada setiap halaman
12	*Menu bar*	*Dropdown*	Menampilkan menu yang terdapat pada halaman administrator

5.2 Implementasi Sistem

Implementasi sistem dapat berjalan setelah mendapatkan keluaran dari proses perancangan. Dalam implementasi sistem dilakukan pencampuran hasil dari proses-proses sebelumnya supaya dapat menjadi sistem yang utuh.

5.2.1 Spesifikasi Sistem

Bagian ini tersusun dari spesifikasi perangkat keras, perangkat lunak dan spesifikasi sistem operasi yang digunakan.

5.2.1.1 Spesifikasi Perangkat Keras

Perangkat keras yang digunakan untuk pengembangan sistem memiliki spesifikasi yang terdapat pada Tabel 5.6.

Tabel 5.7 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Tipe	ASUS GL53VD
Prosesor	Intel Core i7 – 7Th
RAM	16384MB
Penyimpanan	1000GB

5.2.1.2 Spesifikasi Perangkat Lunak

Perangkat lunak yang digunakan untuk pengembangan sistem memiliki spesifikasi yang terdapat pada Tabel 5.7.

Tabel 5.8 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Operation System	Windows 10
Document Editor	Microsoft Word 2016
Diagram Editor	Visual Paradigm 16.3
Text Editor	Visual Studio Code 2019
Programming Language	PHP, Python 3.9
Framework	Codeigniter 3, Bootstrap 3, Materialize
Web Service	XAMPP
Database	MySQL

5.2.2 Implementasi Basis Data

Implementasi basis data akan ditunjukan dengan berkaitan dengan perancangan basis data. Implementasi data pada aplikasi ini menghasilkan satu table baru.

5.2.2.1 Implementasi Data Tabel Cluster

Implementasi dari perancangan tabel cluster ditampilkan dalam bentuk *Data Deifinition Language* (DDL) pada tabel 5.9.

Tabel 5.9 Implementasi Data Tabel *Cluster*

Implementasi data tabel cluster	
1	CREATE TABLE 'cluster' (
2	'tanggal' DATETIME NULL,
3	'clusters' bigint(20) NULL,
4	'id' int(11),
5	ALTER TABLE cluster ADD PRIMARY KEY (id),
6	ALTER TABLE cluster ADD FOREIGN KEY (id) REFERENCES
7	members(id),
8	CREATE INDEX idx_clusters ON cluster (clusters)
9	)

5.2.3 Implementasi Kode Program

Implementasi kode program berkaitan dengan perancangaan komponen. Algoritma *pseudocode* yang telah dituliskan pada bab perancangan komponen akan diimplementasikan kedalam bahasa pemrograman.

5.2.3.1 Implementasi Kode Program Method Member

Pada tabel 5.10 terdapat algoritma *method* member yana berada pada *controller* Authentication. *Method* ini bertujuan untuk melakukan *login* oleh aktor. Algoritma ini akan menangkap inputan aktor berupa *username* dan *password*. *Username* dan *password* tersebut kemudian disimpan dalam satu *variable* untuk dilakukan pemeriksaan dengan data yang telah tersimpan pada *database*. Setiap menekan tombol login, sistem akan merekam tingkah tersebut dalam tabel *log*. Jika *username* atau *password* salah, maka akan mengarahkan kembali pada halaman *login*. Jika benar, maka data *lastlogin* pada member akan diperbaharui kemudian sistem menyimpan *session* yang berisi id *user*, *fullname user*, *username*, dan *cluster user* kemudian mengarahkan pada halaman profil.

Nama *class*: Authentication

Nama *method*: member

Tabel 5.10 Implementasi Kode Program *Method* Member

Source code method member	
1	public function member()
2	{
3	$username = $this->input->post('username');
4	$password = $this->input->post('password');

```
5               $member = $this->m_member->login($username,
6   md5($password));
7               $clusters = $this->m_cluster->get($member['id']);
8               //Log Behavior
9               $this->m_log->add($username, "Login");
10              if (!$member) {
11                  redirect('login');
12              } else {
13                  $this->m_member->lastlogin($member['id']);
14                  $this->session->set_userdata('uid',
15  $member['id']);
16                  $this->session->set_userdata('fullname',
17  $member['fullname']);
18                  $this->session->set_userdata('username',
19  $member['username']);
20                  $this->session->set_userdata('clusters',
21  $clusters['clusters']);
22                  //$this->session->set_userdata('color',
23  $cluster['color']);
24                  redirect('profile');
25              }
26      }
```

5.2.3.2 Implementasi Kode Program Method Clustering

Pada Tabel 5.11 terdapat algoritma *method* clustering yang berada pada *controller* cluster. Tujuan dari *method* ini adalah untuk melakukan fungsi *clustering*. Algoritma diawali dengan deklarasi *variable command* yang menyimpan perintah untuk menjalankan *python* dan direktori *script* yang akan dieksekusi. Kemudian *variable* selanjutnya bernama *output* untuk menjalankan variable *command*. Selanjutnya terdapat *variable* data yang berbentuk *array* untuk menyimpan data dari fungsi get. Setelah itu menampilkan halaman mulai dari *header*, *cluster-list*, dan *footer*.

Nama *class*: Cluster

Nama *method*: Clustering

Tabel 5.11 Implementasi Kode Program *Method Clustering*

```
Source code method clustering
1   public function clustering()
2       {
3           $command = escapeshellcmd('python
```

```
4       D:\Xampp\htdocs\codemaniac\application\python\fcm.py');
5           $output = shell_exec($command);
6
7           $data['clusters'] = $this->m_cluster->get();
8           $this->load->view('admin/header');
9           $this->load->view('admin/cluster-list', $data);
10          $this->load->view('admin/footer');
11      }
12  }
```

5.2.3.3 Implementasi Kode Program Proses Clustering

Pada Tabel 5.12 terdapat Algoritma Proses Clustering yang bertujuan untuk mengelompokkan member menjadi tiga *cluster* sesuai dengan tingkah mereka yang telah tercatat pada tabel *log_behavior* pada *database*. Proses ini berjalan menggunakan bahasa pemrograman *python* yang terintegrasi dengan sistem pada *method clustering*. Proses *clustering* berawal dengan melakukan impor modul-modul yang diperlukan kemudian membuat koneksi dengan *database* untuk selanjutnya melakukan ekstraksi data dari database pada tabel member dan *log_behavior*. Data yang sudah terkumpul tadi digabungkan menjadi satu tabel dan dinormalisasi sebelum memulai proses *clustering* menggunakan modul *fuzzy cmeans*. Data yang telah di-*clustering*, kemudian diberi tanggal dan dikirim kembali ke *database* pada tabel *Clusters*.

Tabel 5.12 Implementasi Kode Program *Proses Clustering*

Source code proses clustering	
1	#!/usr/bin/env python
2	# -*- coding: cp1252 -*-
3	
4	import mysql.connector
5	import sys
6	import csv
7	import pandas as pd
8	import numpy as np
9	import datetime as dt
10	from fcmeans import FCM
11	from sklearn.preprocessing import MinMaxScaler
12	from matplotlib import pyplot as plt
13	from sqlalchemy import create_engine
14	
15	conn = mysql.connector.connect(
16	host = "localhost",

```
    user = "root",
    password = "",
    database = "coma"
)
mycursor = conn.cursor()

engine = create_engine(("mysql+pymysql://{user}:{pw}@{host}
/{db}".format(host="localhost", db="coma", user="root",
 pw="")))

#ekstraksi_data
mycursor.execute("SELECT id, username FROM members")
df_members = pd.DataFrame(mycursor, columns=['id',
 'username'])

mycursor.execute("SELECT log_uid, log_process FROM
 log_behavior")
df_log_behavior = pd.DataFrame(mycursor, columns=
['username', 'log_process'])

df_fusion = pd.merge(df_members, df_log_behavior, how
='inner', on='username')
df_fusion.drop('username', inplace=True, axis=1)

x = ['1','2','3','4','5','6','7','8','9','10','11','12'
,'13','14','15','16','17','18']
count = 0
xl = len(x)

for val in x:
    if xl < 20:
        df_fusion[val] = 0
        count += 1

df_fusion.loc[df_fusion['log_process'] == 'Accept request
friend', '1'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Check exercise',
'2'] = 1
```

```
df_fusion.loc[df_fusion['log_process'] == 'Compile exercise', '3'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Confirm challenge', '4'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Detail exercise', '5'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Login', '6'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Logout', '7'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Search friends', '8'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Send request friend', '9'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Show Profile', '10'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Show Challenge', '11'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Show Exercise', '12'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Show Friends', '13'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Submit exercise', '14'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Take challenge', '15'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Take Course', '16'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Take exercise', '17'] = 1
df_fusion.loc[df_fusion['log_process'] == 'Test exercise', '18'] = 1

df_fusion.drop('log_process', inplace=True, axis=1)

df_final = df_fusion.groupby('id').sum()

# normalisasi
df_floated = df_final.astype(float)
data_cluster = df_floated.iloc[:, :]
```

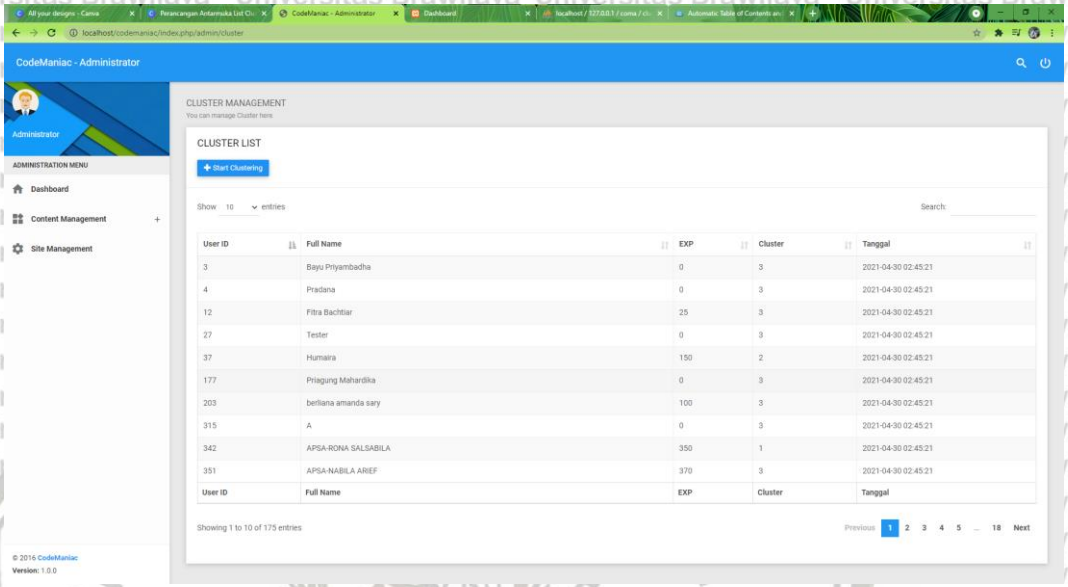
```
dataset = np.array(data_cluster)
data = dataset.tolist()
scaler = MinMaxScaler()
data_minmax = np.around(scaler.fit_transform(data),
decimals=2)

#data_processing
fcm = FCM(n_clusters=3)
fcm.fit(data_minmax)
fcm_centers = fcm.centers
fcm_labels = fcm.u.argmax(axis=1)

df_final['clusters'] = fcm_labels+1
df_final.insert(0, 'tanggal',
pd.to_datetime('now').replace(microsecond=0))

df_final['id'] = df_final.index
df_temp = df_final[['tanggal','clusters','id']]
df_temp.reset_index(drop=True)

df_temp.to_sql('cluster', engine, index=False,
if_exists='replace')

mycursor.execute("ALTER TABLE cluster ADD PRIMARY KEY
(id);")
mycursor.execute("ALTER TABLE cluster MODIFY COLUMN id
INT(11);")
mycursor.execute("ALTER TABLE cluster ADD FOREIGN KEY (id)
REFERENCES members(id);")
mycursor.execute("CREATE INDEX idx_clusters ON cluster
(clusters);")
```

5.2.4 Implementasi Antarmuka

Bagian implementasi antarmuka menjelaskan hasil dari implementasi antarmuka. Implementasi disusun berdasarkan hasil perancangan antarmuka. Pada bab ini terdapat implementasi antarmuka *cluster*, *dashboard*, dan perubahan *interface* yang akan dibahas pada bagian 5.2.4.1 – 5.2.4.3.

5.2.4.1 Implementasi Antarmuka Cluster

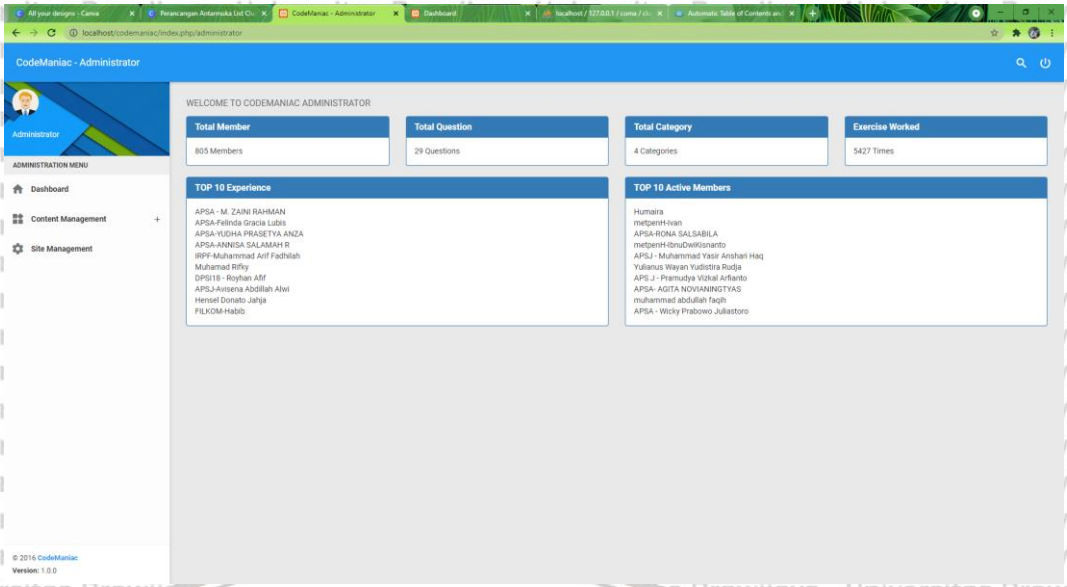
Implementasi antarmuka *cluster* ditampilkan pada Gambar 5.13. Halaman ini tersusun dari tiga bagian utama, yaitu *header*, *sidebar*, dan body. Pada bagian *header* terdapat nama sistem, pencarian, dan *logout*. Pada bagian *sidebar* terdapat menu yang dapat diakses Aktor. Pada bagian *body* terdapat tombol untuk *start clustering*, tabel hasil *clustering* dan *page control*.



Gambar 5.13 Implementasi Antarmuka Cluster

5.2.4.2 Implementasi Antarmuka Dashboard

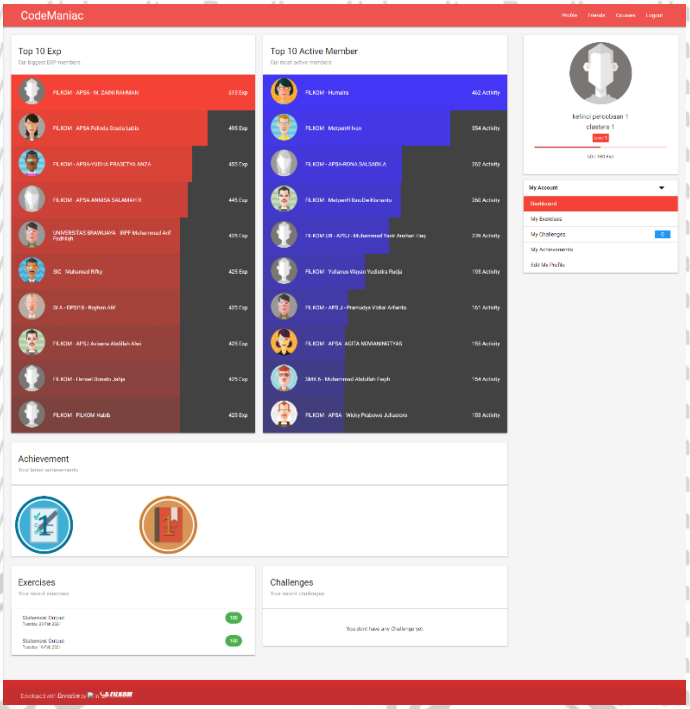
Implementasi antarmuka *dashboard* ditampilkan pada Gambar 5.14. Halaman ini tersusun dari tiga bagian utama, yaitu *header*, *sidebar*, dan *body*. Pada bagian *header* terdapat nama sistem, pencarian, dan *logout*. Pada bagian *sidebar* terdapat menu yang dapat diakses Aktor. Pada bagian *body* terdapat beberapa panel yang menampilkan informasi berbeda disetiap panelnya.



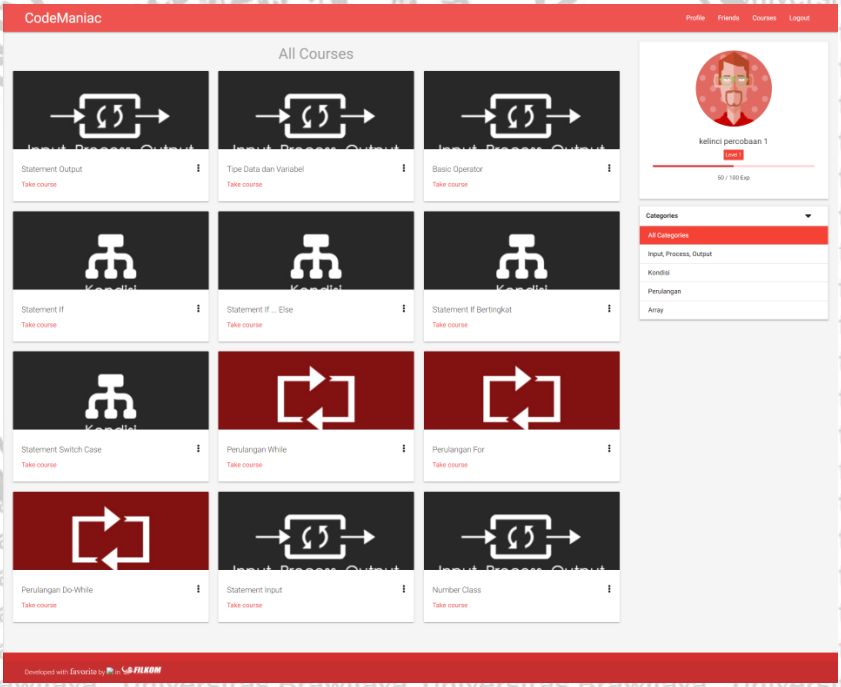
Gambar 5.14 Implementai Antarmuka *Dashboard*

5.2.4.3 Implementasi Antarmuka Perubahan Interface

Sesuai dengan hasil perancangan antarmuka yang telah dijelaskan, hasl implementasi antarmuka dashboard ditampilkan pada Gambar 5.15-5.20. Halaman-halaman ini tersusun dari empat bagian utama, yaitu *header*, *sidebar*, *body*, dan *footer*. Pada bagian *header* terdapat nama sistem, pencarian, dan *logout*. Pada bagian sidebar terdapat menu yang dapat diakses Aktor. Pada bagian body terdapat beberapa panel yang menampilkan informasi berbeda disetiap panelnya.

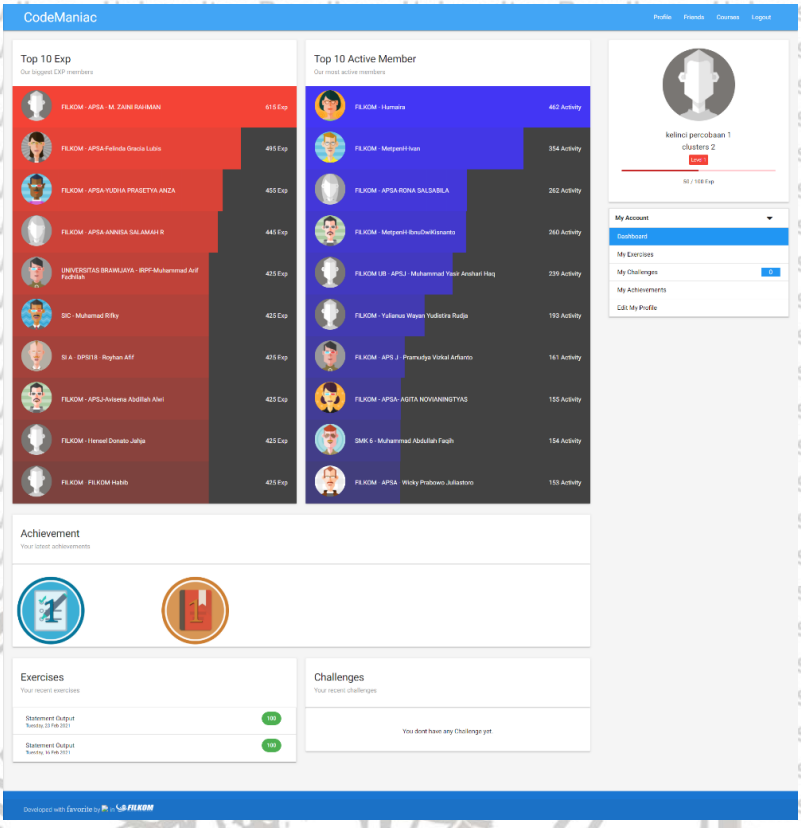


Gambar 5.15 Implementasi Antarmuka Perubahan *Interface* 1

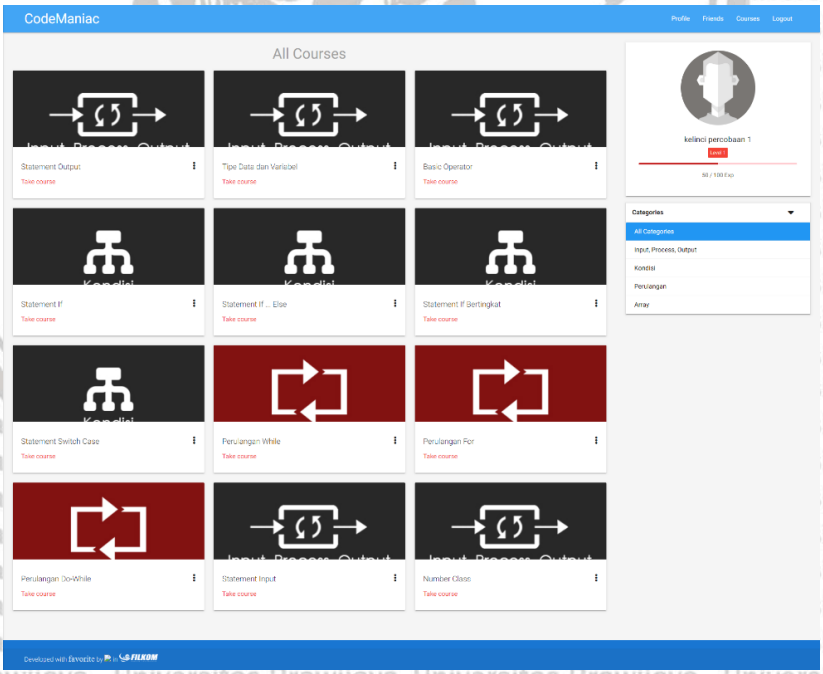


Gambar 5.16 Implementasi Antarmuka Perubahan *Interface* 2

Gambar 5.15 dan Gambar 5.16 menunjukkan hasil implementasi antarmuka apabila seorang member termasuk dalam *cluster* satu. Implementasi antarmuka ini berpengaruh pada warna *header*, *sidebar*, dan *footer* seluruh halaman yang dapat diakses member.

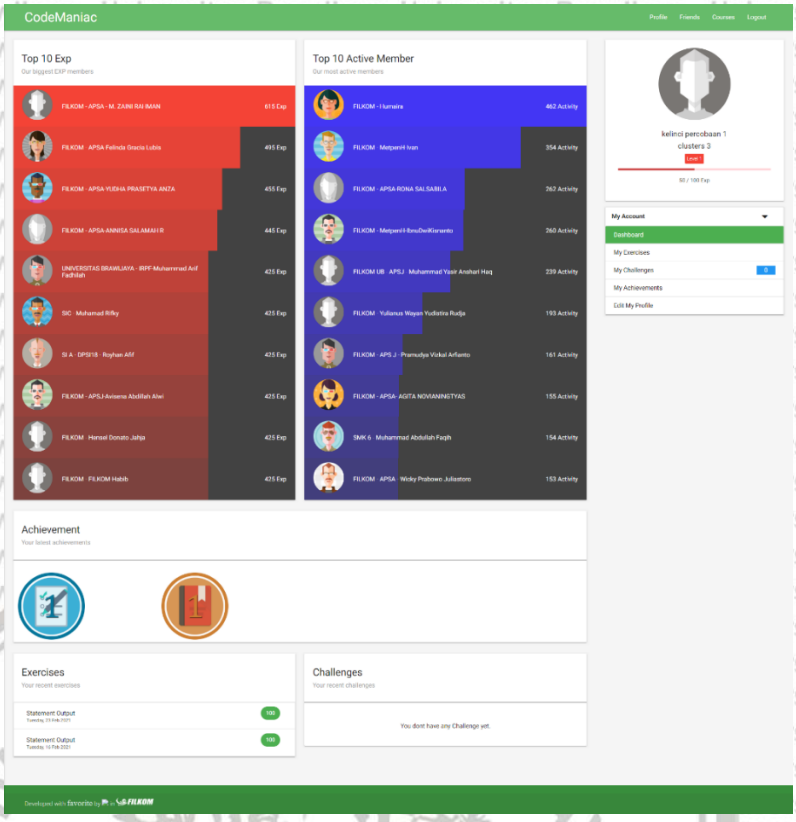


Gambar 5.17 Implementasi Antarmuka Perubahan *Interface* 3

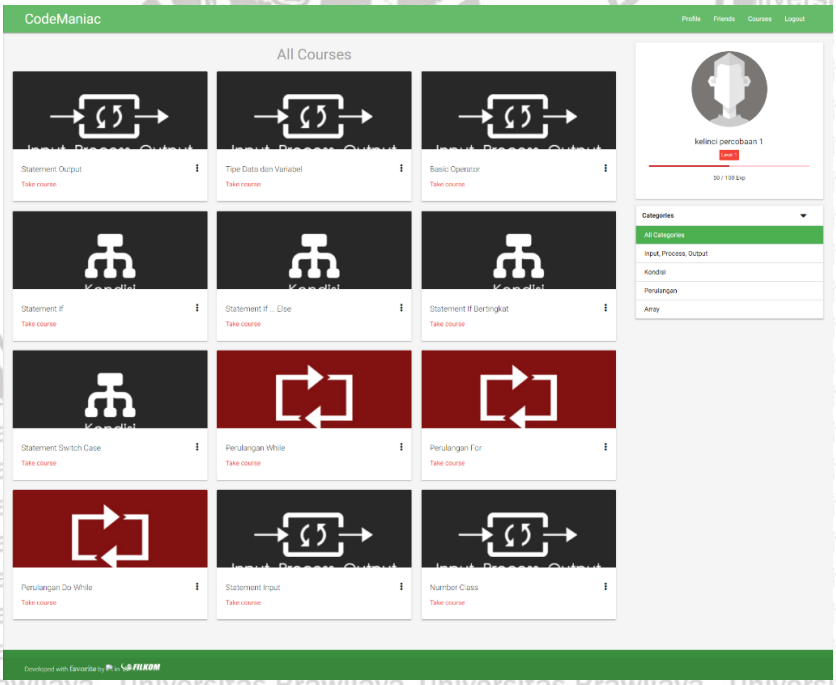


Gambar 5.18 Implementasi Antarmuka Perubahan *Interface* 4

Gambar 5.17 dan Gambar 5.18 menunjukkan hasil implementasi antarmuka apabila seorang member termasuk dalam *cluster* dua. Implementasi antarmuka ini berpengaruh pada warna *header*, *sidebar*, dan *footer* seluruh halaman yang dapat diakses member.



Gambar 5.19 Implementasi Antarmuka Perubahan *Interface* 5



Gambar 5.20 Implementasi Antarmuka Perubahan *Interface* 6

Gambar 5.19 dan Gambar 5.20 menunjukkan hasil implementasi antarmuka apabila seorang member termasuk dalam *cluster* tiga atau belum memiliki *cluster*. Implementasi antarmuka ini berpengaruh pada warna *header*, *sidebar*, dan *footer* seluruh halaman yang dapat diakses member.

BAB 6 PENGUJIAN SISTEM

6.1 Pengujian Unit

Bagian pengujian unit ini, fokus pembahasan ada apa melakukan tes untuk setiap unit terkecil yang telah ditambahkan pada perangkat lunak dengan menggunakan metode *white box testing*. Pada bagian ini akan diuji dua *method* dari *controller* codeigniter dan sebuah fungsi *clustering* pada *script python*. Untuk mengumpulkan *test case* yang akan diujikan menggunakan *basis path testing*.

6.1.1 Pengujian Unit *Method* Member

1. *Pseudocode*

Nama *class*: Authentication

Nama *method*: Member

Tabel 6.1 *Pseudocode Method* Member

Pseudocode method member	
1	Begin
2	Deklarasi variable 'username' = input username; Deklarasi variable 'password' = input password; Deklarasi variable 'member' = menjalankan fungsi Login dari model member dengan parameter (username, md5(password)); Deklarasi variable 'clusters' = menjalankan fungsi get dari model cluster dengan parameter(member['id']); Menjalankan fungsi add dari model log dengan parameter (username, "Login");
3	if (!member) {
4	redirect('login');

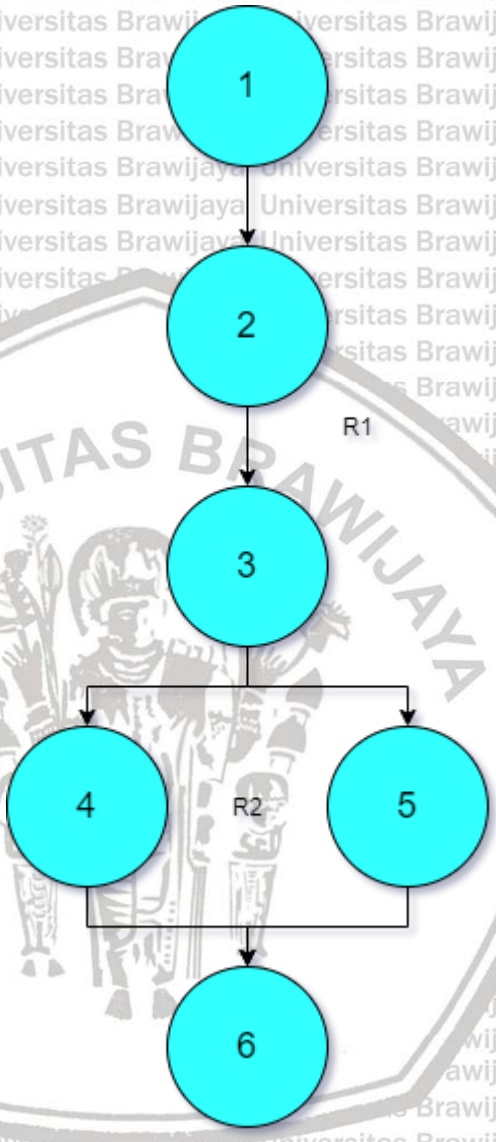
5	} Else {
	Menjalankan fungsi Lastlogin dari model member dengan parameter (member['id']);
	Menyimpan session 'uid' = member['id'];
	Menyimpan session 'fullname' = member['fullname'];
	Menyimpan session 'username' = member['username'];
	Menyimpan session 'clusters' = clusters['clusters'];
	Redirect('profile');
	}

6	end

2. *Basis path testing*

a. *Flow Graph*

Hasil dari penggambaran *flowgraph* berdasarkan *pseudocode* pada pengujian unit *method* member ditunjukkan pada Gambar 6.1.



Gambar 6.1 *Flowgraph Method* Member

b. *Cyclomatic Complexity*

- V(G) = Jumlahi *Region* = 2
- V(G) = Jumlah *edge* – Jumlah *node* + 2 = 6 – 6 + 2 = 2
- V(G) = Jumlah *predicate node* + 1 = 1 + 1 = 2

c. *Independent Path*

- Jalur 1: 1-2-3-4-6

- Jalur 2: 1-2-3-5-6

Jalur independent yang telah didapatkan akan menjadi dasar dari pembuatan kasus uji. Hasil pengujian dapat dilihat pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Unit Method Member

No.	Jalur	Prosedur Uji	Hasil yang Diharapkan	Hasil yang diadapatkan	Hasil
1	1-2-3-4-6	*Method* member() diawali dengan membentuk beberapa *variable* untuk menyimpan inputan *username* dan *password*, kemudian menjalankan fungsi untuk mengecek *username*, *password* dan *cluster*, serta untuk menyimpan *log*. Saat *username* atau *password* salah, maka kembali ke halaman *login*.	Menjalankan *method* member() dan berhasil kembali pada halaman login saat *username* atau *password* salah.	Menjalankan *method* member() dan berhasil kembali pada halaman *login* saat *username* atau *password* salah.	Valid
2	1-2-3-5-6	*Method* member() diawali dengan membentuk beberapa variable untuk menyimpan inputan *username* dan *password*, kemudian menjalankan fungsi untuk mengecek *username*, *password* dan *cluster*, serta untuk menyimpan *log*. Saat *username* atau *password* benar maka akan meyimpan beberapa *variable* dalam *session* dan mengarahkan ke halaman *dashboard*	Menjalankan *method* member() dan berhasil *Login* kedalam sistem.	Menjalankan *method* member() dan berhasil *Login* kedalam sistem	Valid

6.1.2 Pengujian Unit *Method* Cluster

1. *Pseudocode*

Nama *class*: cluster

Nama *method*: clustering

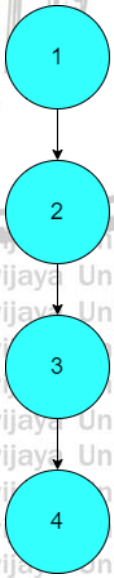
Tabel 6.3 Pseudocode *Method Clustering*

Pseudocode algoritma method cluster	
1	Begin
2	Deklarasi variable command = berisi fungsi escapeshellcmd untuk menjalankan python serta direktori script clustering; Deklarasi variable output = menjalankan fungsi shell_exac dengan parameter command; Deklarasi variable 'data' = menjalankan fungsi get dari model cluster;
3	Mengarahkan kehalaman view 'admin/header'; Mengarahkan kehalaman view 'admin-cluster-list', dan menampilkan data; Mengarahkan kehalaman view 'admin-footer';
4	end

2. *Basis Path Testing*

a. *Flow Graph*

Hasil dari penggambaran *flowgraph* berdasarkan *pseudocode* pada pengujian unit *method cluster* ditunjukkan pada Gambar 6.2.



Gambar 6.2 *Flowgraph Method Cluster*

b. *Cyclomatic Complexity*

- V(G) = Jumlah *Region* = 1
- V(G) = Jumlah *edge* – Jumlah *node* + 2 = 4 – 3 + 2 = 1
- V(G) = Jumlah *predicate node* + 1 = 0 + 1 = 1

c. *Independent Path*

- Jalur 1: 1-2-3-4

Jalur *independent* yang telah didapatkan akan menjadi dasar dari pembuatan kasus uji. Hasil pengujian dapat dilihat pada Tabel 6.4.

Tabel 6.4 Hasil Pengujian Unit Method Cluster

No.	Jalur	Prosedur Uji	Hasil yang Diharapkan	Hasil yang diadapatkan	Hasil
1	1-2-3-4	*Method* cluser() diawali dengan membentuk *variable* untuk menyimpan *command* yang dapat menjalankan *script python* kemudian menjalankan *command* tersebut dan diarahkan pada halaman *cluster*.	*Method* cluser() menjalankan *script python* kemudian diarahkan pada halaman *cluster* dan data *cluster* berhasil diperbarui.	*Method* cluser() menjalankan *script python* kemudian diarahkan pada halaman *cluster* dan data cluster berhasil diperbarui.	Valid

6.1.3 Pengujian Unit *Script Clustering*

1. *Pseudocode*

Tabel 6.5 *Pseudocode Script Clustering*

Pseudocode algoritma script clustering	
1	Begin Impor modul mysql connector; Impor modul sys; Impor modul csv; Impor modul pandas dan simpan sebagai = pd;
2	Impor modul numpy dan simpan sebagai = np; Impor modul datetime dan simpan sebagai = dt; Impor modul FCM; Impor modul MinMaxScaler; Impor modul pyplot dan simpan sebagai = plt; Impor modul create_engine;

3	Deklarasi variable 'conn' untuk menyimpan koneksi dengan database terkait; Deklarasi variable 'mycursor' = conn.cursor; Deklarasi variable 'engine' = untuk mengirim hasil clustering ke database;
4	Mengarahkan mycursor pada id dan username dari tabel members; Deklarasi variable df_members untuk menyimpan data member; Mengarahkan mycursor pada log_uid dan log_process dari tabel log; Deklarasi variable df_log_behavior untuk menyimpan data log; Deklarasi variable df_fusion untuk menggabungkan df_members dan df_log_behavior berdasarkan kolom 'username'; Menghapus kolom 'username' dari df_fusion;
5	Deklarasi variable 'x' berisi integer 1-19; Deklarasi variable 'count' = 0; Deklarasi variable 'xl' = Panjang x;
6	Menambah 18 kolom pada tabel df_fusion; Mengubah value kolom 'log process' menjadi angka 1; Menghapus kolom 'log_process'; Deklarasi variable 'df_final' untuk mengumpulkan df_fusion sesuai dengan kolom 'id';
7	Deklarasi variable 'df_floated' = df_final astype(float); Deklarasi variable 'data_cluster' = df_floated iloc [:, :]; Deklarasi variable 'dataset' = np array(data_cluster); Deklarasi variable 'data' = dataset tolist; Deklarasi variable 'scaler' = MinMaxScaler; Deklarasi variable 'data_minmax' = np around; Deklarasi variable 'fcm' = FCM dengan 3 cluster; Set fcm.fit(data_minmax); Deklarasi variable 'fcm_centers' = fcm centers; Deklarasi variable 'fcm_labels' = fcm u argmax; Set df_final kolom clusters = fcm_label;
8'	Insert tanggal pada df_final; Deklarasi variable 'df_temp' = df_final[tanggal,cluster,id]; Send df_temp to sql; Set mycursor primary key 'id';

```
8'      Set mycursor modify column id = int(11);
        Set mycursor foreign key id cluster pada id members;
        Set mycursor create index pada cluster;
9       end
```

2. *Basis Path Testing*

a. *Flow Graph*

Hasil dari penggambaran *flowgraph* berdasarkan *pseudocode* pada pengujian unit *method cluster* ditunjukkan pada Gambar 6.3.

Gambar 6.3 *Flowgraph Script Clustering*

b. *Cyclomatic Complexity*

- V(G) = Jumlah *Region* = 1
- V(G) = Jumlah *edge* – Jumlah *node* + 2 = 9 – 8 + 2 = 1

- V(G) = Jumlah *predicate node* + 1 = 0 + 1 = 1

c. *Independent Path*

- Jalur 1: 1-2-3-4-5-6-7-8-9

Jalur *independent* yang telah didapatkan akan menjadi dasar dari pembuatan kasus uji. Hasil pengujian dapat dilihat pada Tabel 6.6.

Tabel 6.6 Hasil Pengujian Unit Script Clustering

No.	Jalur	Prosedur Uji	Hasil yang Diharapkan	Hasil yang diadapatkan	Hasil
1	1-2-3-4-5-6-7-8-9	*Script clustering* berawal dengan melakukan *import* modul-modul yang diperlukan, kemudian membuat koneksi dengan *database*. Selanjutnya mengambil data dari *database* dan menyiapkan data tersebut sebelum masuk ke perhitungan. Setelah data siap maka proses *clustering* dengan *fuzzy cmeans* dilakukakan dan yang terakhir mengirim kembali hasil *clustering* ke *database*.	*Database cluster* berhasil diperbarui dengan hasil *clustering* terbaru.	*Database cluster* berhasil diperbarui dengan hasil *clustering* terbaru.	Valid

6.2 Pengujian Validasi

Pada bagian ini skenario dan fungsional-fungsional sistem akan digunakan sebagai acuan untuk pengujian validasi. Pengujian ini bertujuan untuk mengecek bahwa kebutuhan fungsionalitas beserta skenarionya selaras dengan hasil implementasi. Pengujian akan menyatakan valid apabila semua kasus uji telah memenuhi kebutuhan yang ada apa definisi kebutuhan dan skenario. Pengujian validasi akan dipaparkan pada Tabel 6.7 sampai dengan Tabel 6.12.

Tabel 6.7 Pengujian Validasi *Use Case Scenario* Melihat Daftar *Cluster*

Kode Kebutuhan	COMA-F-01
Nama Kasus Uji	Melihat daftar *cluster*
Prosedur	1. Masuk pada halaman admin 2. Membuka menu *content management* 3. Sistem menampilkan daftar menu tersebut 4. Memilih opsi *cluster* 5. Sistem menampilkan daftar *cluster*
Hasil yang Diharapkan	Sistem menampilkan halaman cluster yang berisi daftar *cluster*.
Hasil yang Didapatkan	Sistem berhasil menampilkan halaman *cluster* yang berisi daftar *cluster*.
Status	Valid

Tabel 6.7 merupakan hasil pengujian validasi untuk *use case* melihat daftar *cluster*. Pengujian tersebut bernilai valid karena hasil yang didapatkan sesuai dengan hasil yang diharapkan.

Tabel 6.8 Pengujian Validasi *Use Case Scenario* Melakukan *Clustering*

Kode Kebutuhan	COMA-F-02
Nama Kasus Uji	Melakukan *clustering*
Prosedur	1. Membuka halaman *cluster* 2. Menakan tombol ‘’*start clustering*” 3. Hasil cluster berhasil diperbarui dan ditampilkan
Hasil yang Diharapkan	Proses clustering member berjalan dan hasilnya tabel daftar *cluster* menampilkan hasil terbaru.
Hasil yang Didapatkan	Proses clustering member berhasil berjalan dan tabel daftar *cluster* berhasil dinampilkan hasil terbaru.
Status	Valid

Tabel 6.8 merupakan hasil pengujian validasi untuk use case melakukan *clustering*. Pengujian tersebut bernilai valid karena hasil yang didapatkan sesuai dengan hasil yang diharapkan.

Tabel 6.9 Pengujian Validasi *Use Case Scenario* Menampilkan *Dashboard*

Kode Kebutuhan	COMA-F-03
Nama Kasus Uji	Menampilkan *dashboard*
Prosedur	1. Membuka halaman admin 2. Menekan tombol *dashboard*
Hasil yang Diharapkan	Sistem menampilkan halaman *dashboard* yang berisi beberapa informasi.
Hasil yang Didapatkan	Sistem berhasil menampilkan halaman dashboard yang berisi beberapa informasi.
Status	Valid

Tabel 6.9 merupakan hasil pengujian validasi untuk *use* case menampilkan *dashboard*. Pengujian tersebut bernilai valid karena hasil yang didapatkan sesuai dengan hasil yang diharapkan.

Tabel 6.10 Pengujian Validasi *Use Case Scenario Login*

Kode Kebutuhan	COMA-F-04
Nama Kasus Uji	*Login*
Prosedur	1. Membuka halaman *login* member 2. Mengisi *username* dan *password* 3. Menekan tombol *login* 4. Masuk kedalam sistem
Hasil yang Diharapkan	Sistem menampilkan halaman *login*, setelah mengisi *username* dan *password* maka masuk kedalam sistem.
Hasil yang Didapatkan	Sistem berhasil menampilkan halaman login, setelah mengisi *username* dan *password* maka masuk kedalam sistem.
Status	Valid

Tabel 6.10 merupakan hasil pengujian validasi untuk *use case login*. Pengujian tersebut bernilai valid karena hasil yang didapatkan sesuai dengan hasil yang diharapkan.

Tabel 6.11 Pengujian Validasi Use Case Scenario Login Alternatif 1

Kode Kebutuhan	COMA-F-04
Nama Kasus Uji	*Login*
Prosedur	1. Membuka halaman *login* member 2. Mengisi *username* dan *password* 3. Menekan tombol *login* 4. Kembali menampilkan halaman *login*
Hasil yang Diharapkan	Sistem menampilkan halaman *login*, setelah mengisi *username* dan *password* maka sistem akan tetap menampilkan halaman *login*.
Hasil yang Didapatkan	Sistem berhasil menampilkan halaman *login*, setelah mengisi *username* dan *password* maka sistem akan tetap menampilkan halaman *login*.
Status	Valid

Tabel 6.11 merupakan hasil pengujian validasi untuk *use case login* alternatif satu. Pengujian tersebut bernilai valid karena hasil yang didapatkan sesuai dengan hasil yang diharapkan.

Tabel 6.12 Pengujian Validasi Use Case Scenario Perubahan Antarmuka

Kode Kebutuhan	COMA-F-05
Nama Kasus Uji	Perubahan antarmuka
Prosedur	1. Guest berhasil masuk kedalam sistem.
Hasil yang Diharapkan	Sistem menampilkan antarmuka sesuai dengan *cluster* member, jika termasuk *cluster* satu maka antarmuka berwana merah, *cluster* dua akan menjadi warna biru, dan *cluster* tiga atau belum mendapatkan cluster akan berwarna hijau.
Hasil yang Didapatkan	Sistem berhasil menampilkan antarmuka sesuai dengan *cluster* member, jika termasuk *cluster* satu maka antarmuka berwana merah, *cluster* dua akan menjadi warna biru, dan *cluster* tiga atau belum mendapatkan *cluster* akan berwarna hijau.
Status	Valid

Tabel 6.12 merupakan hasil pengujian validasi untuk *use case* perubahan antarmuka. Pengujian tersebut bernilai valid karena hasil yang didapatkan sesuai dengan hasil yang diharapkan

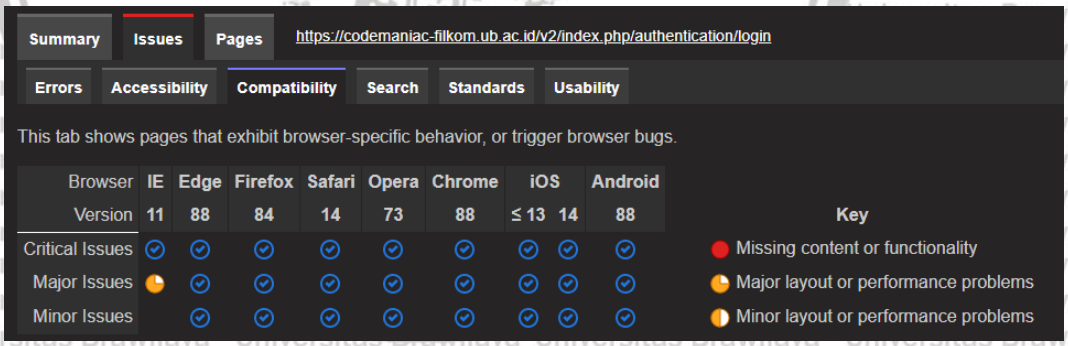
6.3 Pengujian Kompabilitas Aplikasi Web

Tahap pengujian ini dilakukan berdasarkan kebutuhan non-fungsional yang telah dibahas sebelumnya. Pengujian kompabilitas ini akan menggunakan perangkat lunak SortSite yang mempunyai fitur untuk melakukan uji kompabilitas. Hasil pengujian ini akan ditampilkan pada Tabel 6.13.

Tabel 6.13 Pengujian Kompabilitas

Kode Kebutuhan	COMA-NF-01
Nama Kasus Uji	Perubahan Kompabilitas
Prosedur	1. Sistem dapat berjalan pada beberapa jenis *browser* seperti Microsoft Edge, Mozilla Firefox, Google Chrome dan Safari.
Hasil yang Diharapkan	Sistem berjalan dengan baik pada Microsoft Edge, Mozilla Firefox, Google Chrome dan Safari.
Hasil yang Didapatkan	Sistem berhasil berjalan dengan baik pada Microsoft Edge, Mozilla Firefox, Google Chrome dan Safari.
Status	Valid

Pengujian kompalbilitas menggunakan bantuan aplikasi SortSite. Aplikasi ini membantu pengujian kompabilitas dengan menjalankan sistem Codemaniac pada berbagai macam *browser*. Hasil pengujian kompabilitas dengan menggunakan SortSite dapat dilihat pada Gambar 6.4.



Gambar 6.4 Hasil Pengujian Kompabilitas

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil dari setiap proses dalam sisten ini, maka dapat ditarik kesimpulan sebagai berikut:

1. Proses analis kebutuhan yang telah dilakukan menghasilkan lima kebutuhan fungsional dan satu kebutuhan non-fungsional. Pada proses ini juga dihasilkan pemodelan kebutuhan yang berisi *use case digram* dan *use case scenario.*
2. Proses perancangan yang telah dilakukan menghasilkan arsitektur sistem yang berisi *sequence diagram* dan *class diagram* untuk pendekatan berorientasi objek serta *data flow diagram* untuk pendekatan prosedural. Dalam bagian ini juga dihasilkan peracangan komponen, perancangan database dan perancangan antarmuka.
3. Pada proses implementasi yang telah dilakukan menghasilkan spesifikasi sistem, implementasi sistem, implementasi kode program, implementasi basis data dan implementasi antarmuka dengan mengikuti hasil perancangan.
4. Pada proses pengujian yang telah dilakukan tiga pengujian unit dan lima pengujian validasi dan satu pengujian kompabilitas.

7.2 Saran

Saran yang perlu diberikan untuk pengembangan lebih lanjut lagi dari sistem codemaniac ini antara lain adalah:

1. Pengembangan fitur *clustering* dengan memanfaatkan disiplin ilmu komputasi cerdas supaya hasil clustering yang didapatkan menjadi lebih efektif dan efisien.
2. Pengembangan perubahan antarmuka yang lebih kompleks lagi, sehingga perubahan yang dapat sistem tampilkan menjadi lebih bervariasi serta sesuai dengan minat pengguna.

DAFTAR REFRENSI

Bastari, D. I., Pradana, F. & Priyambadha, B., 2017. Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Desember, Volume 1, pp. 1493-1499.

Brusilovsky, P., Kobsa, A. & nejdl, w., 2007. *The Adaptive Web.* 1 ed. Berlin: Springer.

Das, P., C, J. R. & Sajeev, G. P., 2017. Adaptive web personalization system using splay tree. *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017,* Volume 2017, pp. 1582-1587.

Dhupia, B. & Alameen, A., 2019. Adaptive eLearning System: Conceptual Framework for Personalized Study Environment. *Communications in Computer and Information Science,* Volume 1075, pp. 334-342.

Fei, X., Xie, Y., Tang, S. & Hu, J., 2021. Identifying click-requests for the network-side through traffic behavior. *Journal of Network and Computer Applications,* Volume 173.

Hermansyah, F., 2020. *BBC.* [Online] Available at: https://www.bbc.com/indonesia/majalah-53385718 [Accessed 12 september 2020].

James C. Bezdek, R. E. W. F., 1984. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences,* 10(2–3), pp. 191-203.

Marquezan, C. C. et al., 2018. *Adaptive Future Internet Applications: Opportunities and Challenges for Adaptive Web Services Technology.* 1 ed. Hershey: IGI Global.

Pradana, F., Bachtiar, F. A. & Priyambadha, B., 2019. Penilaian Penerimaan Teknologi E-Learning Pemrograman Berbasis Gamification Dengan Metode Technology Acceptance Model (TAM). *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* Volume 6, pp. 163-168.

Pressman, R. S., 2010. *Software Engineering A Practioner's Approach.* 7th ed. New York: McGraw-Hill.

Priyambadha, B., Pradana, F. & Bachtiar, F. A., 2018. PENGGALIAN PERILAKU PEMAIN DALAM PENENTUAN TIPE PERMAINAN. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* Volume 7, pp. 765-772.

Priyambadha, B., Pradana, F. & F. A. B., 2020. PENGGALIAN PERILAKU PEMAIN DALAM PENENTUAN TIPE PERMAINAN. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* 7(4), pp. 765-772.

Prof. Ir. Tian Belawati, M. P., 2020. *Pembelajaran Online (BNBB).* 2 ed. Tangerang Selatan: Universitas Terbuka.

Singh, P. B., n.d. *Data Flow Diagram and its Importance,* Lucknow: Department of Computer Science University of Lucknow.

Sommerville, I., 2011. *Software Engineering.* 9th ed. London: Addison-Wesley.

Tavangarian, D. et al., 2004. Is e-Learning the Solution for Individual Learning?. *Electronic Journal of e-Learning,* 2(2), pp. 273-280.

Urh, M., Vukovic, G., Jereb, E. & Pintar, R., 2015. The Model for Introduction of Gamification into E-learning in Higher Education. *Procedia - Social and Behavioral Sciences,* Volume 197, pp. 388-397.

Verma, A., Khatana, A. & Chaudary, S., 2017. A Comparative Study of Black Box Testing and White Box Testing. *International Journal of Computer Sciences and Engineering,* 5(12), pp. 301-304.

Zakaria, F., 2008. Dynamic profiling of EEG during zeizure using fuzzy information space. *PhD Thesis, Faculty of Science, UTM Malaysia.*